



Universal Machine Learning Methods for Detecting and Temporal Anchoring of Events

Vom Fachbereich Informatik
der Technischen Universität Darmstadt
genehmigte

Dissertation

zur Erlangung des akademischen Grades Dr.-Ing.

vorgelegt von
Nils Fabian Reimers
geboren in Wildeshausen

Tag der Einreichung: 14. Februar 2018

Tag der Disputation: 3. Mai 2018

Referenten: Prof. Dr. Iryna Gurevych, Darmstadt
Prof. Dr. Gerhard Weikum, Saarbrücken
Prof. Dan Roth, Ph.D., Pennsylvania, USA

Darmstadt 2018

D17

Please cite this document as

URN: urn:nbn:de:tuda-tuprints-81634

URL: <https://tuprints.ulb.tu-darmstadt.de/id/eprint/8163>

This document is provided by tuprints,
E-Publishing-Service of the TU Darmstadt

<http://tuprints.ulb.tu-darmstadt.de>

tuprints@ulb.tu-darmstadt.de



This work is published under the following Creative Commons license:
Attribution – Non Commercial – No Derivative Works 4.0 International

<https://creativecommons.org/licenses/by-nc-nd/4.0>

Abstract

Event detection has a lot of use-cases, for example summarization, automatic timeline generation or automatic knowledge base population. However, there is no commonly agreed on definition what counts as an event or how events are expressed in text. As a consequence, many different definitions, annotation schemes and corpora have been published, often focusing on specific applications. For a new application, there is a high chance that new data must be annotated and that a machine learning approach must specifically be trained and tuned for this new dataset.

Instead of a system that works well for one specific dataset, we are interested in a universal learning approach that can be used for a wide range of event detection tasks. In this thesis, we analyze an architecture that is based on bidirectional long short-term memory networks (BiLSTM) and conditional random fields (CRF). The BiLSTM-CRF architecture was successfully used by other researchers for sequence tagging tasks and is a strong candidate for the task of event detection. However, besides numerous hyperparameters, researchers have also published various modifications and extensions of this architecture. These parameters and design choices can have a big impact on the performance and selecting them correctly can make the difference between mediocre and state-of-the-art performance. Which parameters and design choices are of relevance is not clear. This leads to a slow adaptation of the approach to new datasets and requires expert experiences and sometimes brute force search to find optimal parameters. This situation is especially unfavorable for event detection where datasets are often application specific.

In order to accelerate the adaptation to new tasks, we provide an extensive evaluation of the BiLSTM-CRF architecture and its individual components and parameters. We identify which parts are relevant for achieving a good performance and which parameters are important to tune for specific tasks. We derive a standard configuration for the architecture that worked well for various tasks. We then show that the BiLSTM-CRF architecture with the proposed default configuration achieves strong results on different event detection tasks.

In most applications, we are not only interested to know that an event happened, but also need to know when it happened. Different methods for annotating temporal information for events have been proposed. In an annotation study we show that the existent annotation schemes have major drawbacks in providing temporal information for events, at least for news articles. Existent schemes provide insufficient temporal information for the majority of events. This is due to the limitation of the annotation scope to only one sentence or two neighboring sentences. As we show in an annotation study, the relevant temporal information for an event can be several sentences apart from the event mention. We developed a new annotation scheme that addresses short-comings of previous schemes and which requires about 85% less annotation effort. Still, it provides better temporal information for events in a document.

While the new scheme requires less human effort, it creates new challenges for automatic event time extraction systems. Existent schemes can be modeled as a pair-wise

classification task, but this is no longer possible for the new scheme. Instead, the whole document must be considered and information from different parts of the document must be merged together. We propose an automatic system that uses a decision tree with convolutional neural networks as local classifiers. The neural networks consider the whole document. The final label is derived step-wise, with different branching options. Compared to state-of-the-art systems, the developed architecture significantly improves the accuracy for event time extraction on our annotated data. Further, it generalizes well to other datasets and tasks. Without adaption, it improved the F_1 -score for the task of automatic event time line generation for the SemEval-2015 Task 4 by 4.01 percentage points.

The final part of the thesis addresses the evaluation of machine learning approaches. Comparing approaches is a major driving force in our research community, which tries to improve the state-of-the-art for tasks of interest. The question arises how reliable our evaluation methods are to spot differences between approaches. We investigate two evaluation setups that are commonly found in scientific publications and which are the de-facto evaluation setups for shared tasks. We show that these setups are unsuitable to compare learning approaches. This introduces a high risk of drawing wrong conclusions. We identify different sources of variation that must be addressed when comparing machine learning approaches and discuss difficulties of addressing those sources of variations.

Zusammenfassung

Die Erkennung von Ereignissen besitzt viele Anwendungsszenarien, beispielsweise Textzusammenfassung, automatisierte Generierung von Zeitlinien oder die automatisierte Erstellung von Wissensdatenbanken. Allerdings existiert keine weithin akzeptierte Definition, was eigentlich ein Ereignis ist, stattdessen gibt es viele unterschiedliche Definitionen, Annotationsschema und Datensätze. Oftmals zielen diese Definitionen und Datensätze auf spezifische Anwendungsszenarien ab. Dies bedeutet aber, dass für neue Anwendungen oftmals eine neue Definition geschaffen werden muss. Anschließend müssen Daten annotiert werden und ein lernendes System muss auf diesen Daten trainiert werden.

Aufgrund dessen sind wir an Lernverfahren interessiert, die nicht nur auf einem Datensatz gut funktionieren, sondern universell für das Erkennen von Events eingesetzt werden können. Daher analysieren wir in dieser Doktorarbeit eine Architektur, die auf *bidirectional long short-term memory networks* (BiLSTM) und *conditional random fields* (CRF) basiert. Die BiLSTM-CRF Architektur wurde bereits erfolgreich für unterschiedlichste Anwendungen aus dem Bereich *Sequence Tagging* verwendet und ist damit ein vielversprechender Ansatz für die Erkennung von Events. Ein Nachteil der BiLSTM-CRF Architektur ist die hohe Anzahl an Hyperparametern und die hohe Anzahl an konzeptionellen Erweiterungen der Architektur, die von unterschiedlichsten Forschungsgruppen publiziert wurden. Diese Parameter und Designentscheidungen können einen großen Einfluss auf die Performance des Systems haben und es ist nur wenig bekannt, wie die Parameter korrekt zu setzen sind. Dies führt zu einem hohen Aufwand wenn man die Architektur auf einen neuen Datensatz anwenden möchte, da unzählige Parameter und Parameterkombinationen ausprobiert werden müssen. Dies ist besonders kritisch bei der Erkennung von Ereignissen in Texten, da unterschiedlichste applikationsspezifische Datensätze existieren.

Um Aufwand der Adaption für neue Datensätze zu reduzieren, führen wir eine umfassende Analyse der BiLSTM-CRF Architektur durch. Wir identifizieren, welche Parameter und Komponenten der Architektur wichtig für das Erzielen einer guten Performance sind. Darauf aufbauend präsentieren wir eine Standardkonfiguration, die gut funktioniert für eine hohe Anzahl an Datensätzen. Für diese Konfiguration zeigen wir dann, dass diese auch gut für verschiedene Ereignis-Erkennungs-Probleme funktioniert.

In den meisten Anwendungen möchte man nicht nur erkennen, dass ein Ereignis beschrieben wird, sondern man möchte ebenfalls wissen wann dieses Ereignis passiert ist. Es existieren verschiedene Methoden um zeitliche Informationen in Texten zu erfassen und eine Verbindung zu den beschriebenen Ereignissen herzustellen. Wie wir aber in einer Annotationsstudie zeigen, besitzen die existenten Annotationsverfahren, zumindest bei Nachrichtenartikeln, große Nachteile. Existente Annotationsverfahren liefern für einen Großteil der Ereignisse nicht die vom Benutzer gewünschten zeitlichen Informationen. Das Problem existenter Annotationsverfahren ist, dass diese den Annotationsumfang auf denselben bzw. auf benachbarte Sätze beschränken. Zeitliche Informationen für ein Ereignis, dass außerhalb liegt, kann oftmals nicht berücksichtigt werden. Wie wir aber zeigen, kann eine große Anzahl an Sätzen zwischen

dem Ereignis und der zeitliche Informationen liegen. Wir entwickelten deswegen ein neues Annotationsverfahren, welches die Nachteile existenter Verfahren adressiert und zeitgleich 85% weniger Annotationsaufwand erfordert.

Während dieses Annotationsverfahren mit weniger Aufwand für die Annotatoren verbunden ist, stellt es automatisierte Verfahren vor neue Herausforderungen. Existente Annotationsschemata lassen sich als paarweise Klassifikation zwischen dem Ereignis und der zeitlichen Information modellieren. Mit dem neuen Annotationsverfahren ist dies nicht mehr möglich. Stattdessen müssen automatisierte Verfahren das gesamte Dokument betrachten und entscheiden, welche Teile im Text relevant sind. Um diese Herausforderungen zu lösen, präsentieren wir einen Entscheidungsbaum, der in den Knoten *convolutional neural networks* verwendet. Diese neuronale Netzwerke arbeiten auf dem gesamten Textdokument und erzeugen Schrittweise die zeitliche Information für jedes Ereignis im Text. Im Vergleich zu anderen automatisierten System arbeitet das präsentierte System deutlich präziser. Ebenso generalisiert es gut auf neue Daten und Anwendungen. Wir evaluierten es, ohne Anpassung auf den SemEval-2015 Task 4 Datensatz zur Erzeugung automatischer Zeitlinien. Dabei konnte es eine Verbesserung von 4.01 Prozentpunkten erzielen im Vergleich zu anderen Verfahren.

Der letzte Teil der Doktorarbeit beschäftigt sich mit der Evaluation von Lernverfahren. Der Vergleich von Verfahren ist eine treibende Kraft in unserer Forschungsgemeinschaft, die stets versucht, neue und bessere Methoden zu entwickeln. Hierbei entsteht die Frage, wie gut unsere Evaluationsmethoden sind? Wir untersuchen zwei Evaluationsmethoden, die besonders oft in wissenschaftlichen Arbeiten verwendet werden, und zeigen für diese, dass sie ungeeignet sind um Lernverfahren zu vergleichen. Die Schwächen der Evaluationsmethoden führen zu einer hohen Gefahr, dass falsche Schlussfolgerungen gezogen werden. Wir identifizieren verschiedene Faktoren, die die Performance von Lernverfahren beeinflussen, und die in der Evaluationsmethode adressiert werden sollten.

Contents

1	Introduction	1
1.1	Research Questions	4
1.2	Contributions	4
1.3	Publication Record	7
1.4	Thesis Organization	8
2	The Concept of Events	11
2.1	Events in Philosophy	11
2.2	Events in Language	12
2.2.1	TimeML	13
2.2.2	ACE, Light ERE and Rich ERE	14
2.2.3	FrameNet	16
2.3	Existent Event Corpora	16
2.3.1	TimeBank and TimeML Based Corpora	19
2.3.2	ACE and ERE Corpora	21
2.3.3	Further Corpora	27
2.4	Conclusion	28
3	Event Detection using a BiLSTM-CRF Architecture	31
3.1	BiLSTM-CRF Architecture for Sequence Tagging	32
3.2	Configurable Parameters of the BiLSTM-CRF Architecture	35
3.3	Benchmark Datasets	38
3.4	Evaluation Methodology	40
3.5	Evaluation Results	42
3.5.1	Word Embeddings	45
3.5.2	Character Representation	46
3.5.3	Optimizers	48
3.5.4	Gradient Clipping and Normalization	50
3.5.5	Tagging Schemes	52
3.5.6	Classifier - Softmax vs. CRF	53
3.5.7	Dropout	56
3.5.8	Going deeper - Number of LSTM-Layers	58
3.5.9	Going wider - Number of Recurrent Units	59
3.5.10	Mini-Batch Size	61
3.6	Discussion of Evaluation Results	62
3.7	Evaluation on Event Detection Tasks	64
3.8	Conclusion	67
4	Temporal Anchoring of Events	69
4.1	Previous Annotation Work	71
4.1.1	TLINK Based Annotations	72
4.1.2	Time as Event Argument	74

4.2	Document-Wide Event Time Annotation Scheme	76
4.3	Annotation Study	77
4.3.1	Inter-Annotator-Agreement	78
4.3.2	Disagreement Analysis	78
4.3.3	Measuring Partial Agreement	78
4.3.4	Annotation Statistics	79
4.3.5	Most Informative Temporal Expression	79
4.3.6	Comparison of Annotation Schemes	81
4.4	Automatic Event Time Extraction	83
4.5	System Architecture	85
4.5.1	Event Time Extraction using Trees	86
4.5.2	Local Classifiers	87
4.5.3	Baselines	92
4.6	Experimental Setup	93
4.7	Experimental Results	93
4.7.1	System Performance	94
4.7.2	Error Analysis	96
4.7.3	Ablation Test	97
4.7.4	Event Timeline Construction	98
4.8	Conclusion	99
5	Challenges in Evaluating Machine Learning Approaches	103
5.1	Evaluating Learning Approaches vs. Models	105
5.2	Evaluation Methodologies Based on Single Model Performances	106
5.3	Empirical Study: Comparing Methods Based on Single Model Per- formances	108
5.4	Why Comparing Single Model Performances is Insufficient	113
5.5	Sources of Variation	116
5.5.1	Internal Randomness	117
5.5.2	Train, Development and Test Samples	118
5.5.3	Random and Not So Random Class Noise	119
5.6	Evaluation Methodologies Based on Score Distributions	120
5.7	Hyperparameters	124
5.8	Conclusion	127
6	Summary	131
	Appendix	139
A	Guidelines for Annotating Event Time Values	139
	List of Figures	143
	List of Tables	145
	Bibliography	158

Chapter 1

Introduction

Storytelling is central to human existence, and it is common to every known culture (Flanagan, 1992; Boyd, 2009). Stories are used to make sense of our world and to share that understanding with others. Stories revolve around connected events, which can be real or imaginary. Events can be found in all forms of human creativity including speech, literature, theatre, journalism, film, video games and music as well as in some drawings, sculptures, and photographs.

Events are key to journalism. Journalism focuses on the production and distribution of reports on the interaction of events, facts, and ideas. Often, the focus is on new events that are relevant to society. Those new events are usually embedded in a broader context, for example by connecting them to previous events or by discussing possible future events. The occurrence of an event can significantly influence decisions we make and a significant amount of our communication, either verbally or in written form, revolves around events.

Millions of news items reporting on events are published per day (Agerri et al., 2014) and generate a burden for individuals and organizations to keep up with the latest developments. This creates a high demand for automatic event detection and extraction systems. Automatic event systems can be used for information retrieval, for summarization, e.g. by identifying key events in a complex story, or for question answering. Facts in knowledge bases are often based on events, e.g., the birth and death date and place of a famous person, and the automatic population of knowledge bases can highly benefit from high-quality event extraction (Surdeanu, 2013). The purpose of an event extraction system can be summarized to extract the information about “*who did what to whom and perhaps also when and where*” (Jurafsky and Martin, 2009).

Extracting this information from a document can be difficult. An event can be described in many different forms, and the information can be scattered across a sentence or the document. Some information might not explicitly be stated and complex inference is needed. Further, it is possible that some information, for example, the time or the place, is only vaguely specified or not specified at all in the document.

It can be challenging to decide what counts as an event. Text does not only report

about things that actually happened and that are clearly observable. It can also report about abstract, generic, imaginary, conditional, hypothetical, uncertain, or negative events. Further, the distinction between states and events can be difficult (Kim, 1993). This led to many different definitions, annotation guidelines, and corpora. Often, those guidelines and corpora focus on a specific use case, for example by defining only specific types of events.

Without a common annotation scheme for events, no uniform out-of-the-box system for event detection can exist, at least as of today. New applications often require the annotation of data and training of a machine learning approach.

Developing a machine learning system for a new task can be time-consuming and tuning it might require expert experiences. This is especially the case when the approach is sensitive to its parameters or when hand-engineered features must be developed. Hence, a universal learning approach for event detection, which can easily be trained for new tasks, is desirable. The goal would be to have an easy to apply approach for new event detection and extraction tasks that achieves a good performance with no or minimal refinements from the developer.

In this thesis we focus on event detection that can be formulated as a sequence tagging task (cf. chapter 2 for the discussion how events are expressed in a text). The BiLSTM-CRF approach has been shown to work well as a universal learning approach for many sequence tagging tasks (Huang et al., 2015; Ma and Hovy, 2016; Lample et al., 2016). However, the approach consists of many different parameters and many extensions for this approach have been published, that add little twists to it. It is unclear, which parameters and design choices are relevant for a good performance and which parameters must be tuned. Even though the BiLSTM-CRF approach is a rather universal approach for sequence tagging, applying it to new tasks might be difficult due to a large number of parameters and design choices.

Tuning irrelevant parameters or implementing irrelevant design choices can cost a lot of time when adapting the approach to a new task. Hence, in this thesis, we want to identify which parameters and design choices of the BiLSTM-CRF architecture are relevant for achieving a good performance. Further, we want to study which parameters must be tuned for a task and which parameters work well with a certain default configuration. We then evaluate this architecture for the task of event detection. The BiLST-CRF architecture is described in chapter 3.

Event detection is usually the first step in automatic event systems. In further steps, connected information to the event like the participants, the place, or the time are extracted, event coreferences are identified, or the relevance of the event is judged. Which further steps are performed, depends on the specific application.

One crucial and complex task for automatic event systems is the temporal anchoring of events. This step is required for example to detect event coreferences, for the automatic generation of timelines, or for populating knowledge bases. However, when an event happened is often not explicitly stated in documents. Instead, we infer the timeframe when an event happened from the temporal order, from causalities, and from general knowledge. For example in the following news item:

“January 23rd, 2008. Heath Ledger, 28, whose breakthrough role was in the movie Brokeback Mountain, was found dead yesterday in an apartment in SoHo. The chief police spokesman, Paul J. Browne, said the police did not suspect foul play.”

Even though it is not explicitly stated, the reader can infer that Heath Ledger’s role in *Brokeback Mountain* was before his death and after his birth. Further, it can be inferred that the statement of the chief police spokesman was given after the dead body was found and before the publishing of the article.

In [chapter 4](#) we study how events can be anchored in time. We analyze existent annotation schemes and show that those are unable to anchor the majority of events in news articles in time. We then develop a new scheme and perform an annotation study. Compared to other annotation schemes, the scheme provides a more precise temporal anchoring for events at a lower annotation effort.

While the developed scheme is simple for humans, it poses new challenges for automatic approaches. In [section 4.4](#) we present these challenges and propose a decision tree with neural networks as local classifiers to address them. We demonstrate that this approach works well on the annotated corpus and also generalizes well to the task on automatic timeline generation from the SemEval-2015 Task 4 dataset.

Comparing machine learning approaches for tasks we are interested in and concluding which approach is more accurate, is fundamental to our NLP research community. A typical evaluation method in the NLP community is to train and tune approaches on some part of the labeled data, and then to compare the performance scores on unseen test data. A significance test is used to check if the difference in performance might stem from the finite test sample size. When the difference on the test set is significant, the conclusion is drawn that one approach is better (more accurate) for that task than the other approach.

In [chapter 5](#) we show that statistically significant differences on the test set for the evaluation setup must not be due to a better learning approach. There is a high risk that this difference is due to chance. The described evaluation setup does not address randomness introduced by the non-deterministic behavior of the training process. Further, it neglects the problem that test scores are not monotone with development scores and unluckily selecting a model with high development, but low test score can alter drawn conclusions. For two recent papers by [Ma and Hovy \(2016\)](#) and [Lample et al. \(2016\)](#) we show that different conclusions are drawn if the approaches are re-trained with changing sequences of random numbers.

We identify three sources of variations that can affect the comparison of approaches: The internal randomness of the approach, the selection of the datasets, and class noise. An evaluation setup should not be influenced by these sources of variation, however, addressing those can be difficult. Instead of comparing approaches based on individual scores, we propose the comparison of score distributions. We formulate different methods to analyze score distributions and study their ability to compare learning approaches.

1.1 Research Questions

This thesis addresses the following three research questions:

RQ1 Universal Learning Approach for Event Detection

What is understood as an event depends on the use case. Many different definitions, annotation guidelines, and datasets have been published, each following own rules what counts as an event in a text. Hence, instead of a model that works well for one dataset, we are interested to identify a universal learning approach that is suitable for various datasets. The BiLSTM-CRF approach has been successful as a universal learning approach for many sequence tagging tasks. For this approach, we study if it is applicable to the task of event detection. Further, we study which parameters and design choices of the approach are responsible for achieving a good performance. This research question is addressed in [chapter 3](#).

RQ2 Automatic Temporal Anchoring of Events

Events are linked to the temporal dimension and knowing not only that an event happened, but also when it happened, is critical for many use cases. We want to investigate if current approaches can provide sufficient details for the temporal anchoring of events. We start with analyzing if humans are capable of anchoring events in time with a good agreement. Then, we analyze if existent temporal annotation schemes are sufficient for the temporal anchoring of events. Finally, we investigate how an automatic approach can be designed to solve the difficult task of anchoring events in time. This research question is addressed in [chapter 4](#).

RQ3 Evaluation of Machine Learning Approaches

Developing new approaches that are more accurate than previous approaches is a major driving force in our research community. However, how do we decide that a new approach is better than previous approaches? We study if our existent evaluation methodologies can reliably identify which approach is more accurate for a given task. Further, we study which sources of variations can impact the outcome of our experiments and how to address these to ensure that the drawn conclusions are correct. This research question is addressed in [chapter 5](#).

1.2 Contributions

The contributions in this thesis for the research questions are the following:

RQ1 Universal Learning Approach for Event Detection

- We present the BiLSTM-CRF architecture as a universal learning approach for event detection ([chapter 3](#)). The architecture has many tunable hyperparameters, and different variations and extensions for this architecture have been published. We evaluate which parameters and design choices are important for the performance by testing more than 50,000 configurations on common NLP sequence tagging and event detection tasks. We show that only a few parameters are important for tuning. We derive a default configuration that

works well for many diverse sequence tagging tasks. Hence, we expect that this configuration also works well for new sequence tagging and event detection tasks.

RQ2 Automatic Temporal Anchoring of Events

- We show (section 4.1) that the mainly used annotation schemes to anchor events in time fail to provide temporal information for the majority of events. The ACE and ERE standards for events only link temporal information to an event if it is in the same sentence. However, this is only the case for 19.8% of the events in the ACE 2005 dataset. Temporal links (TLINKs) that define the relationship between two events or an event and a temporal expression are usually restricted to relations within the same sentence or within neighboring sentences. Extending the relation to longer distances can be difficult as the number of possible relations grows quadratic. We show that for 58.7% of the events in the TimeBank-Dense Corpus (Cassidy et al., 2014) the needed temporal expression to anchor the event in time cannot be extracted using TLINKs. Even after taking transitivity into account, 21.4% of the Single Day Events cannot be temporally anchored and for 22.7% only a less precise anchoring is possible.
- We develop a new annotation scheme to anchor events in time (section 4.2). Using a defined format, annotators provide the temporal anchoring for all events in a document. The annotation effort for this scheme is linear with the number of events, and, in comparison to a dense TLINK annotation, it is 85% lower. Further, the annotation scheme introduces a concept to annotate the begin and end point for events that last longer than a day (multi-day events). The information on the duration of an event is missing in many other annotation schemes.
- We performed an annotation study on the TimeBank Corpus (section 4.3). The annotation study showed that the temporal expression that defines when an event happened could be several sentences apart from the event mention. Further, it shows that the proposed annotation scheme can be performed efficiently and with a good agreement between annotators. In comparison to dense TLINK annotations, it provides a temporal anchor for each event in a document, and the complete context of the document is taken into account for annotating this temporal anchor. The study showed that for 7.3% of the events it is necessary to infer new dates that are not explicitly stated in the document. Those inferred dates can be the result of semantic inference or world knowledge.
- Existent systems for extracting temporal information for events usually only work on one or two neighboring sentences. This was also due to the lack of training and evaluation data. Extending existent systems to the proposed annotation scheme is not possible due to the fundamental differences in the schemes. Instead, we develop a new approach based on a decision tree that uses convolutional neural networks in it nodes as local classifiers (section 4.5). We show that this approach can extract the new event time annotations. We

show that the approach generalizes well by applying it to the SemEval-2015 Task 4 dataset on automatic timeline generation, where it achieves, without adaptation, state-of-the-art performance.

RQ3 Evaluation of Machine Learning Approaches

- We show (section 5.3) that comparing two machine learning approaches based on the performance of individual models is not possible and significance tests lead to wrong conclusions. We show that training the same network twice can result in large variances in the performance as the network converges to different minima. It was previously known that different minima generalize differently. However, this fact is often neglected when presenting new approaches in our field. For the two recent publications from Lample et al. (2016) and Ma and Hovy (2016) we show that the conclusions change when the provided implementations are executed multiple times with changing sequences of random numbers. For a recent BiLSTM-CRF architecture and seven common NLP sequence tagging tasks, we show that the variance based on the sequence of random numbers is multiple times larger than what is perceived as a significant difference for those tasks.
- We show (section 5.3 and section 5.4) that there is a high risk that statistically significant differences in shared tasks are due to chance and not due to a better learning approach for that task. We show this empirically for seven common NLP sequence tagging task. Further, we proof (section 5.4) that the discovered issue affects any significance test for the usual setup of shared tasks. The test score is a finite approximation of the true performance on the whole data distribution. A significance test checks if two models would not perform differently on the complete data distribution given the performance on the test set. We show that it is not only important to account for in the finite size of the test set, but it is to the same degree important to account for the finite size of the development set. However, usually less attention is spent on the creation of the development set, and in many cases, it is substantially smaller than the test set. In conclusion, we show that learning approaches cannot be compared based on the performance of individual models and there is a high risk that (statistically significant) differences are due to chance.
- We discuss different sources of variation (section 5.5) for machine learning approaches and how to address those in an evaluation (section 5.6). We show that the internal randomness of approaches can be addressed easily in an evaluation by training multiple models with different random sequences. The other sources of variations are much more difficult to address and would require that more labeled data is available. Further, we discuss the challenge that is introduced by hyperparameters (section 5.7).

1.3 Publication Record

Different parts of this thesis have been previously published in international peer-reviewed journals and conferences. Parts of these publications have been reused in this thesis. In the following, we list the publications and link those to the respective chapters. Further, we state whether verbatim quotes from the publications are to be expected.

- In *GermEval-2014: Nested Named Entity Recognition with Neural Networks* (Reimers et al., 2014), published at the KONVENS conference, we presented a deep neural network architecture for nested named entity detection for German, which was ranked 2nd in a shared task. For this architecture, we trained and published one of the first available word embeddings for German.¹
- In *Event Nugget Detection, Classification and Coreference Resolution using Deep Neural Networks and Gradient Boosted Decision Trees* (Reimers and Gurevych, 2015) we adapted the architecture from (Reimers et al., 2014) for the task of event detection on the NIST TAC KBP 2015 events dataset. For the 2015 shared task on event detection, the system was placed first among 14 systems. The architecture is publicly available.²
- In *Task-Oriented Intrinsic Evaluation of Semantic Textual Similarity* (Reimers et al., 2016a), published at COLING, we demonstrated issues with the intrinsic evaluation of Semantic Textual Similarity (STS) measures. We showed that the performance in the commonly used evaluation setup does not correlate with the performance for actual tasks. We proposed alternative evaluation methods that are more suitable to evaluate the quality of STS measures.
- In *Temporal Anchoring of Events for the TimeBank Corpus* (Reimers et al., 2016b), published at ACL, we demonstrate that the existent temporal annotations based on TLINKs in the TimeBank Corpus are insufficient for anchoring events in time. TLINKs are only annotated for relations in the same and in neighboring sentences. However, as shown in an annotation study, relevant temporal information for events can be several sentences apart from the event mention. We developed a new annotation scheme and demonstrated the feasibility for the TimeBank Corpus. The dataset is publicly available.³ The study and the new annotation scheme are described in chapter 4. Passages of this publication are quoted verbatim.
- In *Event Time Extraction with a Decision Tree of Neural Classifiers* (Reimers et al., 2018), published in the TACL journal, we presented an automatic approach for the new annotation scheme from (Reimers et al., 2016b). While the proposed scheme is easier for human annotators, it creates several challenges for automatic approaches. The code is publicly available.⁴ The developed

¹ <https://www.ukp.tu-darmstadt.de/research/ukp-in-challenges/germeval-2014/>

² <https://github.com/UKPLab/tac2015-event-detection>

³ <https://www.ukp.tu-darmstadt.de/data/timeline-generation/temporal-anchoring-of-events/>

⁴ <https://github.com/UKPLab/tac12017-event-time-extraction>

system is presented in [chapter 4](#). Passages of this publication are quoted verbatim.

- In *Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging* (Reimers and Gurevych, 2017a), published at EMNLP, we demonstrated that the random seed value for deep neural networks has a significant impact on the performance of the network. We demonstrated for two recent works from Lample et al. (2016) and Ma and Hovy (2016) that conclusions from their paper change if their implementations are re-run with different random seeds. Instead of comparing machine learning approaches with single performance scores, we propose to compare score distributions. The results of this publication are presented in [chapter 5](#) and passages of the publication are quoted verbatim.
- In *Optimal Hyperparameters for Deep LSTM-Networks for Sequence Labeling Tasks* (Reimers and Gurevych, 2017b), we presented a speed-optimized BiLSTM-CRF architecture and used this implementation to study the importance of different hyperparameters and design choices for this architecture based on more than 50,000 training instances. The results are presented in [chapter 3](#) and passages of the publication are quoted verbatim. The code is publicly available.⁵
- In *Why Comparing Best Model Performances Does Not Allow to Draw Conclusions About Machine Learning Approaches* (Reimers and Gurevych, 2018), we presented that the commonly used setup in our field for shared tasks is not able to identify superior learning approaches. There is a high risk that statistically significant performance difference is due to chance and not due to a superior learning approach. The results of this publication are discussed in [chapter 5](#) and passages of the publication are quoted verbatim.

1.4 Thesis Organization

In the following, we give an overview of the content of the chapters in this thesis.

Chapter 2 discusses the term *event*. *Event* is a rather ambiguous term, and different definitions from philosophy exist what an event in the real world is. In linguistics, different definitions exist how events are expressed in written language. The chapter introduces and discusses the most widely used definitions for events. Further, it discusses challenges when trying to define what an event is, namely the presence of negative, future, hypothetical, conditional, uncertain or generic events in text. The chapter finishes with an overview of the most widely used annotation schemes and corpora for events in NLP.

Chapter 3 introduces the BiLSTM-CRF architecture, which has been proven to be successful for NLP sequence tagging tasks. The thesis focuses on event definitions

⁵ <https://github.com/UKPLab/emnlp2017-bilstm-cnn-crf>

where one or multiple words in a sentence, the so-called *event trigger*, indicates the presence of an event. Those definitions can be modeled as a sequence tagging task. However, no universally accepted definition for events exist, and all annotation schemes and datasets focus on certain applications. Hence, instead of designing one specific system for one definition and dataset, we are interested to identify a universal learning approach that works well on a wide range of event detection tasks. The chances are high that such a learning approach will also perform well for new event detection tasks. We then evaluate the importance of hyperparameters and design choices for the BiLSTM-CRF architecture. The BiLSTM-CRF architecture is highly configurable, with many different parameters and design choices. However, little is known which aspects are important to tune. We show that only few parameters are relevant to achieve a good performance. Using the results, we develop a default configuration and demonstrate that the architecture works well for various event detection datasets.

Chapter 4 introduces previous work in the annotation of temporal information for events. As outlined in the chapter, existent annotation schemes fail to anchor events temporally for the majority of events. This is due to restricting the scope of the annotation to the same or neighboring sentences. We perform an annotation study that shows that temporal information for events can be several sentences apart from the event mention. In this chapter, we propose a new annotation scheme that anchors all events in time. It can be performed efficiently and with a good agreement by human annotators. In contrast to previous schemes, annotators take the complete document into account and are allowed to merge temporal information across a document. We then present an automatic system for this new annotation scheme. While the annotation for humans is simple, and more efficient compared to other annotation schemes, it generates several challenges for automatic systems. The annotators took the complete document into account when anchoring an event in time. Hence, automatic systems must consider the complete document. We present a system that is based on a decision tree and in its nodes it applies local classifiers that are based on convolutional neural networks. We demonstrate that this system can take information from the complete document into account. Further, we demonstrate that the system generalizes to the task of automatic timeline generation.

Chapter 5 starts with showing that two commonly used evaluation methodologies in the NLP community are unsuited to identify superior learning approaches. We show in that chapter that conclusions about learning approaches cannot be drawn based on the performance of individual models. We can observe large performance variances that are due to randomness. We continue with describing three sources of randomness that affect the performance of models which should be addressed in an evaluation setup. We show that an evaluation setup that is based on score distributions instead of individual performance scores can address some of the sources of variations. The chapter finishes with the discussion of hyperparameters and the challenge they pose for comparing learning approaches.

Chapter 6 summarizes the main contributions in this thesis and outlines future work.

Chapter 2

The Concept of Events

Detecting events in a text can be highly useful for many applications. However, there is no commonly accepted definition what an *event* is or which information is connected to an event. The Oxford dictionary gives a rather broad definition for the word *event* (Stevenson, 2010):

A thing that happens or takes place, especially one of importance.

This is one definition for events, however, the definition of events is ambiguous and changes from area to area. Further, there is no agreed standard which types of events exist and which information is connected to an event. In this chapter, we discuss how events are defined in philosophy and in linguistics. We then continue with an introduction of the most widely used annotation schemes and corpora for events in NLP.

2.1 Events in Philosophy

According to the Internet Encyclopedia of Philosophy¹, one of the leading theories of events in philosophy is theorized by Kim (1993). In his theory, an event “*implies change [...] in a substance. A change in a substance occurs when the substance acquires a property it did not previously have, or loses a property it previously had.*” If events signal changes, then *states* express static things that stay unchanged. However, the differentiation between *event* and *state* can be difficult. Kim gives the example of having a throbbing pain in the right elbow as something that is hard to classify. Hence, the term *event* often also includes states.

Kim theorized that events are structured and are composed of three things: a substance (or the object of the event), a property it exemplifies, and a time. The event can be written as a triple $[o, P, t]$ with the object o , the property P and the time t . Kim defines two conditions for events:

- *Existence condition:* An event $[o, P, t]$ exists if and only if the object o exemplifies the n-adic property P at time t .

¹ <http://www.iep.utm.edu/events/>, last accessed October 19th, 2017.

- *Identity condition:* $[o, P, t] = [o', P', t']$ just in the case $o = o'$, $P = P'$, and $t = t'$.

According to Kim, events are non-repeatable, i.e., if an object exemplifies the same property at a different time, it forms a new event. Further, events have a spatiotemporal location, i.e., a geographic location where the event happens at a specific time or time frame.

Stating the time t for an event can be difficult. The sentence

Doris capsized the canoe yesterday

can be formalized to the event $[\text{Doris, capsized the canoe, yesterday}]$. However, as pointed out by Davidson (1969), Doris might have capsized the canoe more than once. Hence, this event might be formalized as $\exists t : [\text{Doris, capsized the canoe, } t]$ and t belongs to yesterday. While some actions, like capsizing a canoe, can occur multiple times on a single day, other actions are difficult or unusual to perform more than once in a short amount of time. For example, it is unusual or even illegal to get married twice on the same day.

This example illustrates the difficulty of correctly representing the temporal anchor for an event. But as Davidson points out, it is a mistake to think that the given example refers to a singular event. Even if Doris capsized the canoe multiple times, the statement that she capsized the canoe once remains true.

Davidson (1969) addresses the question when are events identical. He proposes that events are considered identical if and only if they have exactly the same causes and effects. However, this definition has a circularity issue. Given events e_1 and e_2 , these two are identical if all causes and effects are identical. For example, event e_1 was caused by c_1 and event e_2 was caused by c_2 . Event e_1 and e_2 are identical, if their causes are identical. Causes c_1 and c_2 are again events and deciding if these are identical requires to decide whether their effects (e_1 and e_2) are identical, which was the starting question if e_1 and e_2 are identical. Davidson later sets up a second criterion that events are identical if they happen in the same place and at the same time.

2.2 Events in Language

The definition how events are defined in linguistics and NLP depends on the target application. In topic detection and tracking, the term *event* is often used interchangeably with the term *topic* and describes a cluster of real-world events that are expressed in multiple documents, for example, documents on a big athletic tournament like the Olympics (Allan, 2002). In contrast, information extraction often uses a finer grained definition for what is considered an *event* in a text.

Jurafsky and Martin (2009) describe that understanding an event means to be able to answer the question “*who did what to whom and perhaps also when and where*”. The answer to this question can be scattered across a sentence or document, and the same real-world event can be described in various ways.

In the following sections, and in the rest of the thesis, we focus on this finer grained definition for events from information extraction. We introduce the two most influential definitions and annotation schemes for events in NLP: the Time Markup Language (TimeML) and the Automatic Content Extraction (ACE) standard.

2.2.1 TimeML

A widely studied specification for events in NLP is the Time Markup Language (TimeML) (Saurí et al., 2004). TimeML was developed to answer temporally based questions about events and entities, especially in news articles (Pustejovsky et al., 2003). It was motivated by the fact that existing question answering systems at that time were unable to answer questions incorporating a temporal dimension, for example, questions like “*Is Gates currently CEO of Microsoft?*” or “*When did the Enron merger with Dynegy took place?*”. Those questions cannot be answered without taking the temporal properties of events into account. The goal of TimeML is to be “a common meta-standard for the mark-up of events, their temporal anchoring, and how they are related to each other in news articles” (Pustejovsky et al., 2003).

TimeML defines three major concepts: *events*, *temporal expressions*, and *relations* (Pustejovsky et al., 2003; Saurí et al., 2004). An event is considered as a cover term for situations that *happen* or *occur*. Events can be punctual (*John reached the summit*) or last for a period of time (*John walked up a mountain*). Events are generally expressed by tensed or untensed verbs, nominalizations, adjectives, predicative clauses, or prepositional phrases. Predicates that describe *states* or *circumstances* in which something obtains or holds true, are also considered as events. However, only certain states are annotated. The time span for the state *New York is on the east coast* is longer than the focal interest in a typical newswire text and would not be annotated. TimeML only annotates states that 1) are changed over the course of the document, 2) that are directly related to a temporal expression, 3) that are introduced by an action, or 4) that depend on the document creation time. Events are marked up by annotating a representative of the event expressions, usually the head of the verb phrase:

*Israel has been **scrambling** to **buy** more masks abroad.*

Boldface words are the head of the respective verb phrases and are annotated as an event in TimeML. Generic events that describe a certain type of events, but no particular instantiation, are not tagged in TimeML.

TimeML classifies events into seven generic classes.

- **Reporting events** describe actions of a person of an organization declaring something. Examples are *say* or *tell*.
- **Perception events** involve the physical perception of another event. Examples are *see* or *hear*.

- **Aspectual events** are a grammatical device of aspectual predication. Examples are *begin* or *finish*.
- **Intensional action events** introduce another event. Examples are ***trying** to monopolize* or ***investigate** the genocide*, where bold marks the intensional action event and underline marks the introduced event.
- **Intensional states** describe states that refer to alternative worlds. An example is *Russia now **feels** [the US must hold off]*, where bold marks the intensional state and the alternative world is indicated by square brackets.
- **States** describe circumstances in which something obtains or holds true. An example is *He was **CTO** for several years*.
- **Occurrence events** describe everything that happens or occurs in the world. Example are *landed* or *arrived*.

Temporal expressions in TimeML can be points in time, intervals, or durations. They can be fully specified like *June 11th, 1989*, underspecified such as *Monday*, intensionally specified such as *last week*, or a duration like *two years*. Each annotated temporal expression is assigned one of the following types: DATE, TIME, DURATION, or SET. DATE expressions represent calendar dates, TIME expression refers to a time of the day, DURATION describes a duration and SET describes a set of dates.

Besides *events* and *temporal expressions*, TimeML specifies three relation types between two events, two temporal expressions or between an event and a temporal expression. Most notably are temporal links (TLINKs) that specify the temporal order. The intention of TLINKs is to enable temporal ordering of events, and, where possible, to retrieve the calendar date of the event. TLINKs are discussed in greater detail in chapter 4.

TimeML further defines subordination links (SLINK), which are used for context introducing relations, and aspectual links (ALINK) for capturing the relation between an aspectual event and its target event. However, these two relation types received less attention in subsequent research.

2.2.2 ACE, Light ERE and Rich ERE

The annotated events in TimeML are only linked to the temporal dimension. However, there is no linkage to the geographical dimension and no linkage to entities that participated in an event. Further, events are classified only coarsely into seven, mostly syntactical, classes.

In contrast, the Automatic Content Extraction (ACE) standard provides consistent annotations for entities, events, and relations in documents. The development of the standard started in 1999 by NIST and the first version focused on the annotation of entities in English documents. In subsequent years the standard was extended and in 2005 guidelines for the annotation of events for Arabic, English, and Chinese were added (ACE, 2005).

The 2005 guidelines for the annotation of events defines an event as ”*a specific occurrence involving participants. An Event is something that happens. An Event can frequently be described as a change of state.*“ An event is represented by an *event trigger* and *event arguments*. An *event trigger* is the word that most clearly expresses the occurrence of the event. In most cases, it is a verb, and in some cases, it is an adjective or a past participle. *Event arguments* can be attributes of an event or entities that participated in the event. Each argument is characterized by a role that it plays in the event, for example, agent, object, source, or target.

In the sentence

*John was **born** in England*

the word *born* would be marked as the event trigger, *John* as the argument of the person that was born and *England* as the event argument for the birthplace. All three values together form the event.

The ACE guidelines tag only certain types of events. Eight main event types are defined: *life*, *movement*, *transaction*, *business*, *conflict*, *contact*, *personnel*, and *justice*. Each type defines several subtypes, for example *be-born* and *marry* are subtypes of *life*. In total, there are 33 defined subtypes. Note, ACE events do not cover other types of events although they might appear in a text. This is an important distinction to TimeML.

The arguments can be of different forms, e.g. temporal, location, instrument, or purpose. However, even though events are defined as specific occurrences involving participants, no argument is obligatory. The value for arguments are noun phrases within the sentence of the event trigger, i.e., no values outside the sentence are possible. This definition is fairly similar to semantic role labeling which focuses on *who did what to whom, when, where, and how*.

The Light ERE (Entities, Relations, Events) standard was created under the DARPA DEFT program as a lighter alternative of ACE. The goal was making annotations easier and more consistent across annotators (Aguilar et al., 2014). This was achieved by consolidating some of the most problematic annotation type distinctions. The definition and tagging of an event remained similar. Both standards have almost identical event categories. There were only minor changes for the subtypes of the contact and movement event types. In contrast to ACE, Light ERE does not tag negative, future, hypothetical, conditional, uncertain or generic events. The event trigger for ACE is a single word, while Light ERE allows the trigger to be a word or a phrase that instantiates the event (Linguistic Data Consortium, 2013). For Light ERE, only asserted participants in an event are annotated as event arguments.

In a second phase, Light ERE was extended to form Rich ERE (Song et al., 2015). Rich ERE expands the ontology for entities, relations, and events. Further, it introduced the concept of *Event Hoppers* to annotate event coreferences within and across documents. Rich ERE added one new main event type (*manufacture*) that has only a single subtype (*artifact*). Further, it added several new event subtypes to existent event main types. In total, 38 different event subtypes are defined. It re-

versed the decision not to tag negative, future, hypothetical, conditional, uncertain or generic events. In Rich ERE, those events are annotated and a specific attribute, the realis attribute, is set for those events. This is compatible with the event tagging in the ACE standard. Rich ERE also reversed the decision to annotate only asserted participants. Now, participants that might have participated in an event are annotated as well, as it is the case for the ACE standard. While Light ERE required that an event has at least one event argument, Rich ERE allows the annotation of argument free events. Further, Rich ERE permits double tagging of event triggers if those infer multiple events.

2.2.3 FrameNet

FrameNet is based on the theory of frame semantics from Charles J. Fillmore and colleagues (Fillmore, 1976, 1982). It can be understood on the basis of "semantic frames, a description of a type of event, relation, or entity and the participants in it."² The definition of semantic frames in FrameNet is comparable to the definition of events in ACE / ERE. An event in ACE / ERE consists of an *event trigger* and a set of arguments, while a frame in FrameNet consists of frame-invoking words (lexical units) and a set of frame elements that define participants and attributes in a frame.

The relation and attribute types in the ACE / ERE standards can be mapped to FrameNet frames (Aguilar et al., 2014). However, there is a slight distinction between FrameNet and ACE / ERE. FrameNet prioritizes lexicographic and linguistic completeness and frames tend to be much finer grained. As of October 23rd, 2017, FrameNet defined 1223 different frame types, while ACE only defined 33 event types. Note, while a large number of frames are events under the definition that something happens or holds true (states), frames also exist to describe entities or relations and their properties. For example, the *animals* frame is used to capture the characteristics of animals described in a text.

Due to the high structural similarity between FrameNet and ACE, researchers successfully used FrameNet to identify events in ACE (Liu et al., 2016) or retrained frame extraction systems for event detection (Judea and Strube, 2015).

2.3 Existent Event Corpora

An overview of the most important existent corpora for event detection and extraction is given in the following table. A more detailed discussion of these corpora is provided in the following sections.

The corpora do not only differentiate in size or the textual domains but also how events are defined and which information is annotated. Some corpora only annotate

² <https://framenet.icsi.berkeley.edu/fndrupal/WhatIsFrameNet>, last accessed October 23rd, 2017.

event mentions, while others provide information that is connected to the event, for example, a semantic class, event participants, and locations, or event coreference chains. Further, some corpora provide temporal relations (TLINKs) between events and temporal expressions.

Corpus	Size	Description
TimeBank 1.2 (Pustejovsky et al., 2003)	7935 event mentions in 183 documents. 73% of the documents are from the Wall Street Journal, 14% are transcriptions from TV or radio broadcast news and 13% are newswire articles from Associate Press and New York Times. 6418 TLINK annotations.	Annotation based on TimeML. Annotation of events, temporal expressions, and temporal links (TLINKs). Event is defined as a cover term for situations that happen or occur as well as states and circumstances expressed. No annotation of event arguments and no semantic types for events.
TempEval-1 (Verhagen et al., 2007)	6832 event mentions and 5790 TLINK annotations.	Based on TimeBank. Reduced the set of possible TLINK classes and added TLINKs for relations in the same sentence.
TempEval-2 (Verhagen et al., 2010)	5688 event mentions and 4907 TLINK annotations.	Based on TimeBank. Review of all event annotations and addition of TLINKs
TempEval-3 (UzZaman et al., 2013)	11145 event mentions and 11098 TLINK annotations.	Extension of TimeBank corpus. Addition of a new platinum test set and addition of the AQUAINT TimeML Corpus to the training set.
TimeBank-Dense (Cassidy et al., 2014)	1729 event mentions and 12715 TLINK annotations.	Subset of the TimeBank Corpus. Annotation of all TLINKs in the same and in neighboring sentences (dense TLINK annotation).
ACE 2005 (Walker et al., 2005)	5349 event mentions, 9793 event arguments, and 54824 entities in 599 documents. Documents are from newswire, broadcast news, blogs, discussion forums, and conversational telephone speech.	Annotation of entities, event triggers and event arguments for 33 event types. Events from other types are not annotated. No annotation of temporal expressions and temporal relations.

TAC 2015 Event Dataset (Mitamura et al., 2015a)	12976 event mentions in 360 documents. Documents are from newswire and internet discussion forums.	High similarity to the ACE 2005 dataset. Annotation based on the Rich ERE annotation guideline. Annotation of event triggers and event arguments for 38 event types. Events from other types are not annotated. No annotation of temporal expressions and temporal relations.
Richer Event Description (O’Gorman et al., 2016)	95 newswire, discussion forum and narrative text documents containing 8731 event mentions, 1127 temporal expressions, and 10320 entity mentions.	Synthesis of the THYME-TimeML guidelines, the Stanford Event coreference guidelines, and the CMU Event coreference guidelines. Annotation of event triggers and entities, but no annotation of event arguments. Annotation of temporal relations and event coreferences. No semantic types for events.
MEANTIME (Minard et al., 2016)	Annotation of the five first sentences in 120 Wikinews articles on four topics.	Annotation based on the NewsReader guidelines, which defines entities, event, temporal expressions, and relations. Annotation of entity and event coreferences. Annotation of entities was inspired by the ACE 2005 guidelines, and the annotation of events was inspired by TimeML.
ECB (Bejan and Harabagiu, 2010)	Annotation of 1744 event mentions, 339 within-document event coreferences and 208 cross-document event coreferences in 482 news texts from Google News archive on 43 topics.	Focus on coreferences of events.
EECB (Lee et al., 2012)	Same 482 documents as the ECB corpus. 2533 event mentions and 774 event coreference chains.	Extension of the ECB corpus by Lee et al. (2012) by following the OntoNotes guidelines for coreference annotations. Adding annotations for entities and events mentions in partially annotated sentences.

ECB+ (Cybulska and Vossen, 2014)	982 documents and 15003 event mentions. 2319 cross-document event coreference chains. 2205 location and 12677 participant annotations.	Extension of the ECB corpus by Cybulska and Vossen (2014). Addition of 502 new documents and addition of event participants and locations.
-------------------------------------	--	--

Often, the corpora were developed with a specific application in mind and provide only certain information. For example, some corpora provide temporal relations between events, but no event arguments or event coreferences. Table 2.2 gives an overview of the mentioned corpora and which type of information related to events is annotated.

Corpus	Types	Arguments	Coref.	Temporal
TimeBank 1.2	×	×	×	✓
TempEval-1	×	×	×	✓
TempEval-2	×	×	×	✓
TempEval-3	×	×	×	✓
TimeBank-Dense	×	×	×	✓
ACE 2005	✓	✓	×	×
TAC 2015 Event Dataset	✓	✓	×	×
Richer Event Description	×	×	✓	✓
MEANTIME	×	✓	✓	✓
ECB	×	×	✓	×
EECB	×	×	✓	×
ECB+	×	×	✓	×

Table 2.2: Properties of event corpora. *Types*: Semantic type of the event, *Arguments*: definition of event arguments, like participants or location, linked to an event, *Coref.*: event coreferences, *Temporal*: temporal relations between events.

As Table 2.2 shows, no corpus contains all the information we might be interested in. The MEANTIME corpus (Minard et al., 2016) provides annotations for event arguments, event coreferences as well as temporal relations for events. However, with only 597 annotated sentences it is rather small. Further, it does not provide information about the semantic type of an event, which can be critical information for downstream applications.

2.3.1 TimeBank and TimeML Based Corpora

A well studied corpus for event detection is the *TimeBank Corpus*³. The TimeBank Corpus contains 183 news articles that have been annotated using the TimeML spec-

³ <http://www.timeml.org/timebank/timebank.html>

ification (Saurí et al., 2004). According to Pustejovsky et al. (2003), the documents were chosen to cover a wide variety of media sources:

- 134 out of 183 (73%) documents stem from the Wall Street Journal that were published between October 25th, 1989 and November 2nd, 1989.
- 25 documents (14%) are transcriptions of TV or radio broadcast news (ABC, CNN, ea, ed, PRI, VOA), mainly from January to March 1998.
- 24 documents (13%) are newswire articles from Associate Press (AP) and from the New York Times (NYT), mainly from February 1998.

The most frequently used and studied annotations in TimeBank are the annotations for events, temporal expressions and temporal links (TLINKs). The annotation was done in two stages. The first stage was carried out by five annotators and 70% of the documents were annotated. However, all of the annotators participated in the creation of the TimeML annotation scheme. In the second stage, 45 computer science students annotated the remaining 30% of the documents. Statistics on the annotations are provided in Table 2.3.

Documents	183
Tokens	61,418
Events	7935
Temporal Expressions	1414
TLINKs	6418

Table 2.3: Statistics on TimeBank version 1.2

Ten documents of the TimeBank version 1.2 were annotated by two experienced annotators, and those annotations were used to compute inter-annotator agreement.⁴ One annotation served as gold data, and the F_1 -score was computed for the annotations of the other annotator. The inter-annotator agreement is depicted in Table 2.4.

TimeBank IAA	F_1
Events	
span	0.78
class	0.77
Temporal expressions	
span	0.83
value	0.90
TLINKs	
pair	0.55
type	0.77

Table 2.4: Inter-annotator agreement for selective attributes in TimeBank.

⁴ Source: <http://www.timeml.org/timebank/documentation-1.2.html#iaa>, last accessed: October 24th, 2017

It is stated that the low agreement for TLINK annotations is due to the large number of possible pairs, and only salient TLINKs were annotated. However, which relations are considered salient is not specified, and annotators disagreed which relations are important. The issue of selecting the pair for a TLINK annotation is further discussed in chapter 4.

The TimeBank corpus served as a basis for several shared tasks. For the shared task TempEval-1 (Verhagen et al., 2007), the organizers used the event and time annotation verbatim from TimeBank. TLINKs were newly added for this task with a focus on links in the same sentence. However, only a reduced set of relational classes was used. For the shared task TempEval-2 (Verhagen et al., 2010), the task organizers reviewed all event annotations to make sure that those compile with the latest annotation guidelines. Additionally, further TLINKs were added. The task organizers also released datasets for Chinese (about 23,000 tokens), Italian (about 27,000 tokens), French (about 19,000 tokens), Korean (about 14,000 tokens) and Spanish (about 68,000 tokens). For the latest shared task on TimeBank, TempEval-3 (UzZaman et al., 2013), the organizers extended the annotation. A new platinum test set on unseen text (about 6,400 tokens) has been annotated by the organizers, who were experts in this area, resulting in a higher agreement for this platinum corpus (cf. Table 2.5). Further, the organizers added the AQUAINT TimeML Corpus⁵ (about 34,000 tokens) to the training dataset.

TempEval-3 IAA	F_1
Events	
span	0.87
class	0.92
Temporal expressions	
span	0.87
value	0.88

Table 2.5: Inter-annotator agreement for the platinum corpus of TempEval-3 (UzZaman et al., 2013).

A lot of attention received the TLINK annotations. TempEval-1, -2, and -3 mainly focused on adding TLINKs within a sentence. More dense annotations for TLINKs have been applied by Bramsen et al. (2006), Kolomiyets et al. (2012), Do et al. (2012) and by Cassidy et al. (2014). However, while the ratio of TLINKs per event increased, the total number of annotated events decreased. An overview of corpora, that are based on TimeBank, is given in Table 2.6. The annotation work of these authors is discussed in more detail in chapter 4.

2.3.2 ACE and ERE Corpora

The ACE 2005 Corpus⁶ is a multi-lingual corpus that contains annotations for entities, event triggers, event arguments, and relations for the languages English and

⁵ <http://www.timeml.org/timebank/timebank.html>

⁶ <https://catalog.ldc.upenn.edu/ldc2006t06>

Corpus	Events	Temporal Expressions	TLINKs
TimeBank	7935	1414	6418
TempEval-1	6832	1249	5790
TempEval-2	5688	2117	4907
TempEval-3	11145	2078	11098
Bramsen et al. (2006)	627	-	615
Kolomiyets et al. (2012)	1233	-	1139
Do et al. (2012)	324	232	3132
Cassidy et al. (2014)	1729	289	12715

Table 2.6: Statistics for corpora that are based on TimeBank.

Chinese. For Arabic, only entities and relations were annotated.

The selection of documents was driven to provide at least 50 examples of each entity, relation and event type/subtype. Documents were quickly labeled as *good* or *bad* based on the number and type of entities, relations and events. For *good* documents, annotators estimated roughly the number of each type. Eventually, documents were algorithmically selected to maximize the overall count for each type and subtype. However, it was not ensured that 50 examples for each type were provided.

Table 2.9 lists the number of event mentions per event type. The corpus has a strong class imbalance. The *attack* event is the most common event type and accounts for 1543 out of 5349 (29%) event mentions. Some other types are infrequent, for example, there are only two *pardon* events in the corpus. At least 50 examples in the training set are only provided for 20 out of 33 event types.

For English, the annotated corpus consists of 599 documents from various sources and domains. Further, for five out of six document categories there is a temporal split between training and test documents. The English corpus consists of documents from the following domains:

- 18% of the documents are newswire articles from Agence France-Presse, Associated Press, New York Times and Xinhua News Agency. Training documents are from March to June 2003. Test documents are from July to August 2003.
- 38% of the documents are broadcast news from CNN and CNN Headline News. Training documents are from March to June 2003. Test documents are from July to August 2003.
- 10% of the documents are broadcast conversations from CNN CrossFire, CNN Inside Politics, and CNN Late Edition. Training documents are from March to June 2003. Test documents are from July to August 2003.
- 20% of the documents are from various weblogs. Training documents are from November 2004 to February 2005. Test documents are from March to April 2005.
- 8% of the documents are various internet discussion forums. Training documents are from November 2004 to February 2005. Test documents are from

March to April 2005.

- 7% of the documents are from conversational telephone speech. Training and test documents stem both from November to December 2004.

While the number of documents varies between the six domains, the number of words per domain is roughly the same and varies between about 37,000 and 56,000 words. Details on the number of annotated entities, events and event arguments can be found in Table 2.7.

Domain	Documents	Words	Entities	Events	Event Arg.
Newswire	106	48399	11025	1557	3334
Broadcast news	226	55967	1184	3518	2334
Broadcast conv.	60	40415	914	2328	1414
Blogs	119	37897	6547	507	998
Discussion forums	49	37366	6516	719	1043
Speech	39	39845	9933	468	670
Total	599	259889	54824	5349	9793

Table 2.7: Statistics for the ACE 2005 corpus.

All data was annotated by two, independently working annotators. Discrepancies between the two annotators were solved by a senior annotator or a team leader. Mitamura et al. (2015b) state that the inter-annotator agreement on the span of events is at 64.8% F_1 -score and the agreement for the type of the event is at 62.2% F_1 -score.

The Linguistic Data Consortium (LDC) released a corpus annotated with the RichERE Annotation Guidelines version 2.5.1 (Linguistic Data Consortium, 2015) for the Event Detection and Coreference shared tasks at the NIST Text Analysis Conference Knowledge Base Population (NIST TAC KBP) 2015. The annotation process is described by Song et al. (2015). Annotations are provided for three languages: English, Chinese and Spanish. For the English version, 158 annotated documents are provided for training, and 202 annotated documents are provided for the evaluation of systems. The English and Spanish corpus consist of newswire articles as well as posts from discussion forums, while the Chinese Corpus has only posts from discussion forums. An overview of the corpus is provided in Table 2.8.

The documents for this dataset were selected automatically. An automatic event detection system was trained on the ACE corpus and was applied to candidate documents. Those documents were ranked in descending order by the event density, which is defined by the number of event triggers per 1,000 tokens. Song et al. (2015) report that the selected documents are much richer in terms of events compared to a prior approach where no ranking was imposed.

That the selected documents were richer regarding events can be confirmed by comparing the TAC KBP 2015 events dataset with the ACE 2005 dataset. Both datasets annotated roughly the same classes of events. The TAC 2015 dataset has on average 55 event mentions per 1,000 tokens, while the ACE 2005 dataset has only about 21 event mentions per 1,000 tokens.

TAC KBP 2015 Events Dataset	Train	Test
Documents	158	202
Newswire documents	81	98
Discussion forum documents	77	104
Tokens	139,444	98,414
Events	6538	6438
Double tagged events	323	575

Table 2.8: Statistics for the NIST TAC KBP 2015 events dataset.

An overview of the number of events per event type is provided in Table 2.9. While some event types have plentiful examples, several other types have only few examples. For the ending of a business (*Business End Org*), only 13 events in the train set are provided. For the type of *Manufacture* events, only 22 examples are provided. At least 50 examples in the training set are only provided for 28 out of 38 event types. This class imbalance can make it difficult for automatic classifiers to detect infrequent event types.

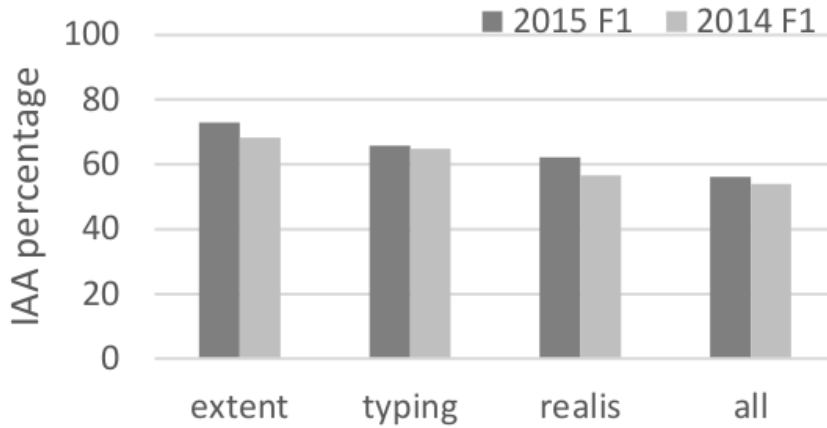


Figure 2.1: Inter-annotator agreement for NIST TAC KBP 2015 events dataset. Image source: (Song et al., 2016)

Figure 2.1 displays the inter-annotator agreement for the 2014 as well as the 2015 version of this dataset. The agreement on the extent of an event trigger without the type is at about 72% F_1 -score. With typing, the agreement is at about 65% F_1 -score. In an own manual re-annotation on a subset of sentences, we achieved an agreement F_1 -score for the extent of the event of 76.57% (Reimers and Gurevych, 2015). We observed an especially low agreement for *contact* events. It was later confirmed that the agreement for *contact* events is fairly low (Figure 2.2) compared to the other events. According to the annotation guidelines, only the first occurrence of speech verbs like *said* or *told* should be tagged if they refer to the same event. However, annotators varied in implementing this rule resulting in inconsistent annotations (Song et al., 2016).

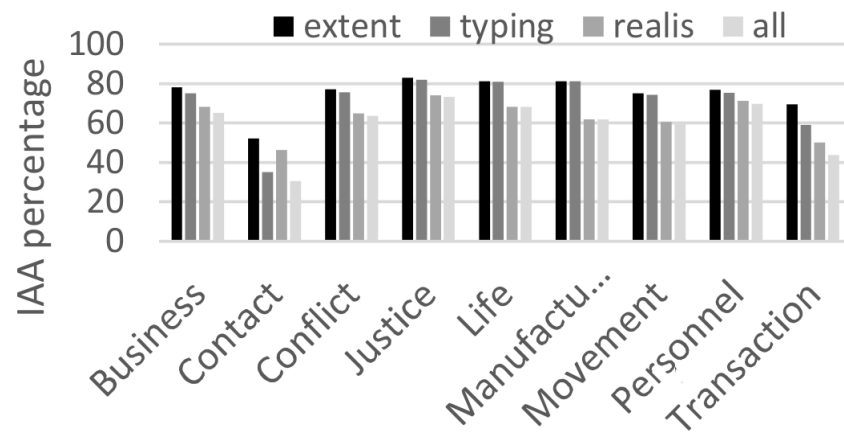


Figure 2.2: Inter-annotator agreement for NIST TAC KBP 2015 events dataset per event type. Image source: ([Song et al., 2016](#))

Type	Subtype	#Events ACE	#Events TAC 2015	
			Train	Test
Business	Declare Bankruptcy	43	33	44
Business	End Org	37	13	6
Business	Merge Org	14	28	33
Business	Start Org	47	18	35
Conflict	Attack	1543	800	591
Conflict	Demonstrate	81	200	149
Contact	Broadcast	-	417	510
Contact	Contact	-	337	587
Contact	Correspondence	-	95	110
Contact	Meet	280	244	272
Contact	Phone-Write	123	-	-
Justice	Acquit	6	30	31
Justice	Arrest-Jail	43	37	69
Justice	Appeal	88	287	348
Justice	Charge-Indict	106	190	155
Justice	Convict	76	222	96
Justice	Execute	21	66	97
Justice	Extradite	7	63	60
Justice	Fine	28	55	45
Justice	Pardon	2	239	51
Justice	Release-Parole	47	73	124
Justice	Sentence	99	144	158
Justice	Sue	76	55	72
Justice	Trial-Hearing	109	196	155
Life	Be Born	50	19	17
Life	Die	598	514	408
Life	Divorce	29	45	49
Life	Injure	142	133	87
Life	Marry	83	76	83
Manufacture	Artifact	-	22	90
Movement	Transport	721	-	-
Movement	Transport.Artifact	-	70	66
Movement	Transport.Person	-	517	439
Personnel	Elect	183	97	71
Personnel	End Position	212	209	291
Personnel	Nominate	12	35	63
Personnel	Start Position	118	77	94
Transaction	Transaction	-	51	63
Transaction	Transfer-Money	198	551	554
Transaction	Transfer-Ownership	127	280	265
		5349	6538	6438

Table 2.9: Label distribution for the ACE 2005 corpus and for the NIST TAC KBP 2015 dataset for event detection. The TAC 2015 dataset has an official split in train and test data. Types that do not exist in a corpus are marked with a dash.

2.3.3 Further Corpora

The Richer Event Description (Palmer et al., 2016) is an annotation scheme which was developed “as a synthesis of the THYME-TimeML guidelines, the Stanford Event coreference guidelines, and the CMU Event coreference guidelines.” It combines event coreference (Pradhan et al., 2007) with THYME temporal relation annotations (Styler et al., 2014), which is an extension of TimeML for clinical notes. The goal is to provide a rich representation of entities, events, times, coreference and partial coreference relations as well as for temporal, causal and subevent relationships between events (O’Gorman et al., 2016). The Richer Event Description corpus contains 95 documents from news data and casual discussion forums, totaling 54,287 tokens, 5,731 events, and 10,320 entities. The inter-annotator agreement for the annotation of events is at 86.1% F_1 -score.

The MEANTIME Corpus (the NewsReader Multilingual Event ANd TIME Corpus) consists of 120 English Wikinews articles on four topics: Airbus and Boeing, Apple Inc., General Motors, Chrysler and Ford, and stock market (Minard et al., 2016). Further, the corpus includes a translation of those articles into Italian, Spanish, and Dutch. The annotation of events is based on the NewsReader guidelines (Tonelli et al., 2014), which defines entities, events, temporal expressions, and numerical expressions as well as relations between those. Further, entity and event coreferences were annotated. The annotation for entities was inspired by the ACE annotation for entities, however, in some cases, it was simplified. The annotation for events was inspired by TimeML. Only the first five sentences were annotated. In comparison to the other corpora, the MEANTIME corpus is rather small with only 597 annotated sentences (13,981 tokens) and 2,096 event mentions for the English version.

The EventCorefBank (ECB) (Bejan and Harabagiu, 2010) is a corpus that focuses on the coreference of events. It consists of 482 news texts from Google News archive on 43 different topics. Events were annotated in accordance with the TimeML specification (cf. section 2.2.1). The authors annotated coreferences for 339 within-document events and 208 cross-document events. The ECB corpus was extended by Lee et al. (2012) by following the OntoNotes annotation guidelines for coreference annotations (Pradhan et al., 2007). They extended the original corpus by annotating entities and events in sentences that were partially annotated.

Cybulska and Vossen (2014) augmented the ECB corpus with 502 new documents to make it more representative of news articles on the web. Further, they added annotations for event participants and locations expanding on the ACE entity subtypes. The participants in an event can be humans, organizations as well as geopolitical entities (GPE), vehicles, and facilities when those are referring to a population or a government. The annotation was performed *event centric*, i.e., participants, locations, and temporal expressions were only annotated if those were part of an event opposed to annotating any participant, location, or temporal expression that is mentioned in a sentence.

2.4 Conclusion

The definition of an *event* in the Oxford dictionary is rather vague and describes *a thing that happens or takes place*. Similar, in philosophy, an event *implies change in a substance* and the distinction between *events* and *states* can be subtle. As a consequence, there is no agreed definition how events are expressed in written language.

Two influential definitions for events are the Time Markup Language (TimeML) (Saurí et al., 2004) and the Automatic Content Extraction (ACE) standard (ACE, 2005). TimeML understands an event as a cover term for situations that *happen* or *occur* as well as *states* and *circumstances* that something obtains or holds true. Events are generally expressed by tensed or untensed verbs, nominalizations, adjectives, predicative clauses, or prepositional phrases.

TimeML uses a broad definition for events and most verbs in a text are events under the TimeML definition. However, events in TimeML are not semantically grouped, and TimeML does not provide annotation guidelines for event participants and event attributes for events (*event arguments*). Hence, corpora, which use only the TimeML annotation guidelines like the TimeBank Corpus (Pustejovsky et al., 2003), miss information on the semantic type of an event and information about the participants in an event. This limits the usefulness of TimeML for downstream information retrieval tasks, as critical information on *who did what to whom* is missing.

The Automatic Content Extraction (ACE) standard (ACE, 2005) focuses on the participants and attributes for events. It defines 33 types of events and for each type it defines roles (event arguments) about participants and attributes. These roles are annotated and can subsequently be extracted from automated systems. This makes the ACE standard much more suitable for downstream information retrieval tasks as the information *who did what to whom* can be extracted. However, the scope of the arguments is limited to the same sentence. If the information for arguments is provided in a different sentence, it is not annotated, and as a consequence, automated systems, that take the complete document into account, cannot be trained and evaluated.

An ongoing point of controversy is the treatment of negative events, uncertain events, hypothetical events, conditional events, future events, and generic events. If a text expresses that a person didn't go to the cinema, should this be annotated as an event in a text? How would such an event be anchored in time and place? Similar with uncertain or conditional events, does the phrase *"if the company files bankruptcy, thousands will lose their jobs"* express that something happens or takes place?

In TimeML and ACE, these events are annotated as they can contain valuable information for readers. However, the annotation of such events can be difficult, and the agreement between annotators is usually lower than for events that describe actual things that happened. Further, as shown in our annotation study in chapter 4, the temporal anchoring for these type of events is especially difficult with a low agreement between annotators. For the Light ERE standard (Aguilar et al., 2014),

which is a lighter alternative of ACE with the goal of making annotations easier and more consistent, it was decided to not annotate those events. However, the extension of Light ERE, called Rich ERE, re-introduced the annotation of negative events, uncertain events, hypothetical events, conditional events, and future events.

In summary, there is no widely accepted definition for marking the expression of events in a text. As shown in Table 2.2, the corpora are rather distinct which information was annotated. If the user is interested in the semantic type of a detected event or wants to know the participants of an event, the TimeML standard is insufficient, as this information is not provided. The ACE standard defines semantic types for events and provides annotations for attributes as well as participants for events. However, the ACE standard is limited to the 33 defined types and events with other types are not annotated and hence are not extracted by automatic systems based on the ACE standard. Further, if an event actually happened or if it is, for example, a generic event can be a crucial distinction for downstream applications. However, this distinction is not always provided in the annotation schemes, and it would be the responsibility of the downstream task to decide if an event actually happened or if the text reports about a negative, uncertain, hypothetical, conditional, or future event.

As there is no generally accepted definition for events, there is no single event detection and extraction model. Instead, depending on the used corpus, trained approaches can be used only for specific applications. For a different application, it might be required that new data must be annotated and that a new approach must be developed. Training of machine learning systems can be a complex task, requiring a lot of time, computational resources, and expertise. Many systems require the tuning of hyperparameters, which is described as a *"black art that requires expert experience, unwritten rules of thumb, or sometimes brute-force search"* (Snoek et al., 2012). Hence, it would be desirable to have a machine learning approach that performs well on various datasets, and that requires only minimal tuning when applied to new datasets.

In the rest of the thesis, we will focus on the following aspects:

1. **Universal machine learning approaches for event detection:** Most published approaches for event detection focused on one corpus and in some cases the approaches rely on corpus specific information, for example, annotated event arguments are used to determine the event type. How these approaches perform on a different corpus, with a different definition for events, is usually not evaluated. We argue that it is desirable to have a universal approach that works well for various event definitions and corpora. Such a universal approach can be easier adapted to new tasks and dataset than corpus-specific approaches. Chapter 3 introduces the BiLSTM-CRF architecture, which performs well for various sequence tagging task. However, so far, it was unclear which aspect of this complex architecture is responsible for achieving a good performance and which parts must be tuned when applied to a new task. Hence, we performed an extensive evaluation of this architecture on various sequence tagging tasks. We identify the parameters that contribute the most to the performance of the architecture. Further, we demonstrate that

this architecture performs well without adaptation on various event detection tasks. Results of this work have been published and presented at EMNLP 2017 (Reimers and Gurevych, 2017a).

2. **Temporal anchoring of events:** Events are inseparably connected to time. However, as chapter 4 shows, existent annotation schemes and systems perform poorly in the task of anchoring events in time. Only a minority of events can be temporally anchored. The issue is not the system, but it is how the event time is annotated or linked to an event in existent annotation standards and corpora. We develop a new annotation scheme that allows the temporal anchoring for all events in a document. We present in chapter 4 an automatic system that allows to anchor events in time. Previous systems only took the sentence and the neighboring sentences of an event into account. The presented system is able to take the complete document into account. The annotation scheme was published and presented at ACL 2016 (Reimers et al., 2016b), the automated system has been published in the Transactions of the Association for Computational Linguistics (TACL) (Reimers et al., 2018).

In our research community, we strive to find new learning approaches that perform better (more accurate) than previous approaches. In the final part of this thesis, we study how reliable our evaluation methodology is to compare learning approaches and which points might affect the conclusions we draw. One commonly used evaluation methodology is to compare the performance of two approaches on unseen test data. A significance test is used to check whether the difference is statistically significant, and if this is the case, the conclusion of a superior approach is drawn. As we will demonstrate, this is an insufficient method to compare approaches. There is a high risk that statistically significant differences on unseen test data are not due to a superior approach, but simply due to chance. The reason is not a flawed significance test, but that this evaluation setup is in general infeasible to decide which approaches is more accurate for a task. We present different sources of variations, that can impact the outcome of our experiments and which should be addressed in our evaluation method.

Chapter 3

Event Detection using a BiLSTM-CRF Architecture

As discussed in [chapter 2](#), no commonly agreed event definition exists. As a result, many different annotation schemes and datasets exist, sometimes tailored for a specific use case. For new use cases, it is likely that a new definition is needed with the consequence that new data must be annotated and eventually a new model must be trained for this particular definition and dataset. Hence, it is desirable to have a universal learning approach that can easily be trained on new datasets and achieves a good performance. The effort for adapting the approach for the new dataset should be reduced to a minimum.

Deep neural networks have been shown to work well on a large set of problems in NLP. They can often be applied with minimal adaptation to new (similar) tasks. [Collobert et al. \(2011\)](#) presented a universal neural network architecture that can be used for various natural language processing tasks including part-of-speech tagging, chunking, named entity recognition, and semantic role labeling. Without task-specific engineering or other prior knowledge, the approach was able to achieve state-of-the-art or near state-of-the-art performance for those tasks. We demonstrated that neural networks can successfully be used to identify events in a text and can work well without task-specific features ([Reimers and Gurevych, 2015](#)).

Event detection, as discussed in [chapter 2](#), can be defined as a sequence tagging task. The *event trigger* is the word or phrase that most clearly expresses the occurrence of an event. In most cases, it is a verb, an adjective or a past participle. The goal of event detection systems is to identify these event triggers in a text and maybe to assign further attributes to the event, for example, a semantic class.

A deep neural network architecture that proved to be useful for sequence tagging tasks is the BiLSTM-CRF architecture ([Huang et al., 2015](#); [Ma and Hovy, 2016](#); [Lample et al., 2016](#)), which achieves state-of-the-art performance for many common NLP sequence tagging tasks.

In this chapter, we introduce this network architecture and different variations that have been published. However, it is unclear which of these variations will perform

best for event detection, and it can be expensive to implement, test and tune all these variations for a new dataset. This bears the risk that time is unnecessarily spent to implement and optimized unneeded options and parameters. As the architecture has a large number of design choices and parameters, there is also the risk that an important aspect is not tuned for a new task.

In this chapter, we provide an extensive evaluation of the different design choices and parameters with respect to their importance for the classification performance. For each design choice and parameter, we estimate the importance for the performance and, if possible, propose a default value that worked well in the evaluated tasks. The results of this study have been published in (Reimers and Gurevych, 2017a) and (Reimers and Gurevych, 2017b).

Using the results of this evaluation, we propose a default configuration for the BiLSTM-CRF architecture. We evaluate this configuration on five event detection tasks and demonstrate that this architecture is in general suitable for detecting events in text.

3.1 BiLSTM-CRF Architecture for Sequence Tagging

Recurrent neural networks (RNNs) are a family of neural networks that operate on sequential data. They take as input a sequence, for example, a sequence of words, and return another sequence, for example, the tags for an NLP sequence tagging task. In theory, RNNs are able to learn long dependencies between the input sequence and the output sequence. However, training recurrent networks can be difficult due to the vanishing and exploding gradient problem (Bengio et al., 1994). The vanishing gradient problem causes that RNNs are most often biased towards the most recent elements in the sequence.

Long Short-Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997a) were designed to cope with the vanishing gradient problem and to be able to take longer dependencies into account. LSTM networks incorporate a memory cell which has been shown in some cases to capture long-range dependencies. The memory cell is controlled by several gates that add, update and delete information from the memory cell.

Huang et al. (2015) demonstrated for part-of-speech tagging, chunking and named entity recognition that LSTM networks can successfully be used for NLP sequence tagging. The proposed architectures are depicted in Figure 3.1. Figure a) depicts a single forward LSTM network that reads in the embeddings for the words in a text and outputs the tags for the task. However, Huang et al. demonstrated that having a second LSTM network, that reads the text from the end to the beginning, is beneficial (depicted in b)). For a given target word that we like to tag, a bidirectional LSTM (BiLSTM) network can consider information from words that appear before the target word as well as information that appears after the target word.

For many sequence tagging tasks, the tags have dependencies. For example, for POS tagging, an article is often followed by an adjective or a noun. For sequence tagging tasks, where segments instead of single words are tagged, a tagging scheme like the BIO scheme can be used. There, the I-tag requires a preceding B-tag. To capture dependencies between tags, Huang et al. proposed a BiLSTM architecture that incorporates a linear chain Conditional Random Field (depicted in c)). They showed that this BiLSTM-CRF architecture can produce state-of-the-art (or close to) accuracy on POS, chunking, and NER datasets.

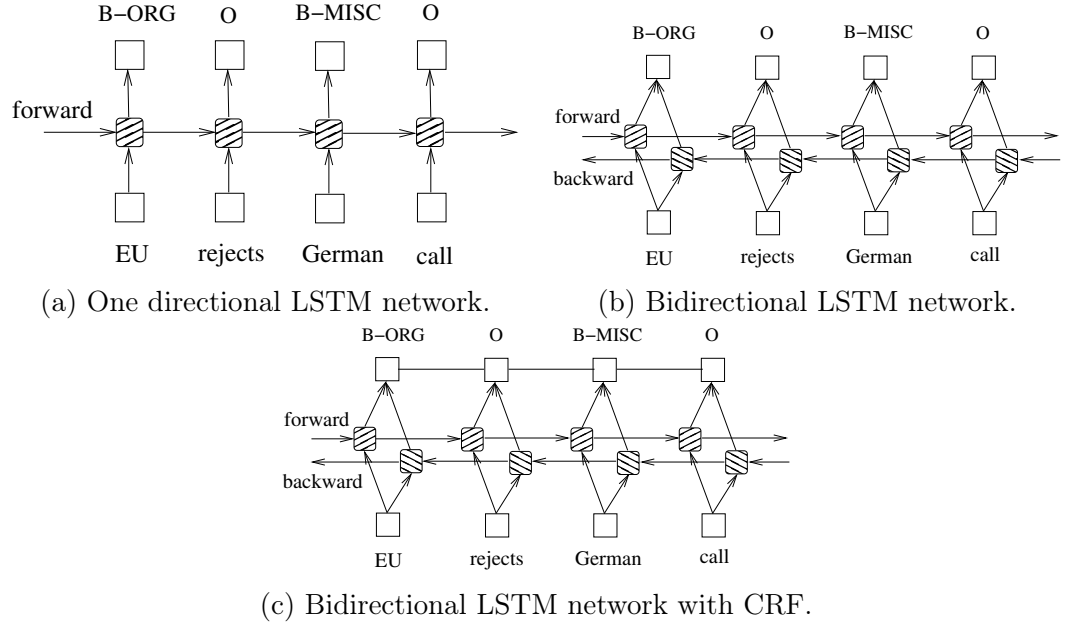


Figure 3.1: Illustration from [Huang et al. \(2015\)](#) on the proposed LSTM architectures for sequence tagging.

[Ma and Hovy \(2016\)](#) and [Lample et al. \(2016\)](#) extended the presented BiLSTM-CRF architecture by adding strategies to derive word representations from the characters of the word. The goal is to extract morphological information that can be used for the classification. Further, such character-based word representations can also be derived for rare words that do not have a pre-computed word embedding.

[Lample et al. \(2016\)](#) map each character in a word onto a randomly initialized character embedding ([Figure 3.2](#)). Those character embeddings are then processed by a BiLSTM network and the final output of the two LSTMs are concatenated and used as a dense character-based word representation. Lample et al. used 25 recurrent units, hence the output is a 50-dimensional vector.

[Ma and Hovy \(2016\)](#) used a similar idea, but instead of using a BiLSTM network, they proposed to use a convolutional neural network ([LeCun et al., 1989](#)) to derive character-based word representations. Each character is mapped to a randomly initialized embedding and a convolution with 30 filters and a filter length of 3 (i.e. character trigrams) is used, followed by a max-over-time pooling to derive one 30-dimensional dense representation for a word.

[Ma and Hovy \(2016\)](#) and [Lample et al. \(2016\)](#) concatenate the character-based word

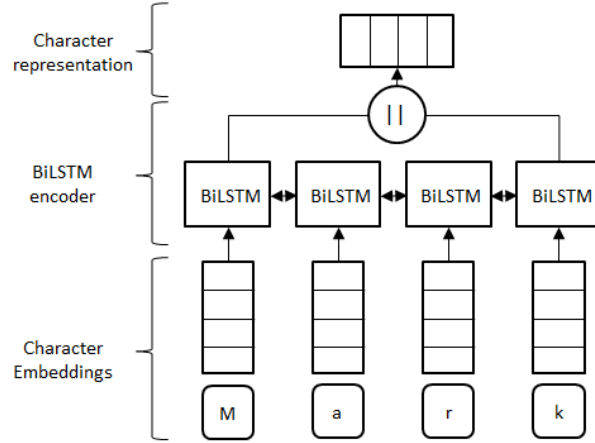


Figure 3.2: Strategy from [Lample et al. \(2016\)](#) to derive character-based word representations.

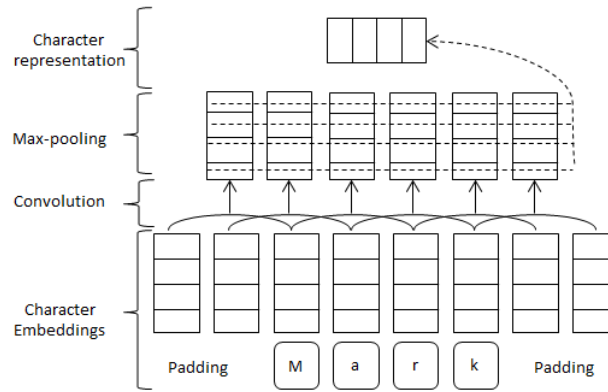


Figure 3.3: Strategy from [Ma and Hovy \(2016\)](#) to derive character-based word representations.

representation with a pre-trained word embedding, e.g. from word2vec ([Mikolov et al., 2013](#)), and feed this to the BiLSTM-CRF architecture from Huang et al.

Based on the published BiLSTM-CRF architectures, we developed our own uniform learning approach which is depicted in [Figure 3.4](#). In comparison to the published implementations, our implementation focuses on flexibility: Many aspects of the architecture are configurable and can easily be replaced by different design choices. This configurability allows to compare different setups and to identify design choices that generalize well across many tasks.

Our architecture maps each word in a sentence to a (pre-trained) word embedding. As word embeddings are often only provided for lowercased words, we add a capitalization feature that captures the original casing of the word. The capitalization feature assigns the label *numeric* if each character is numeric, *mainly numeric* if more than 50% of the characters are numeric, *all lower* and *all upper* if all characters are lower-/uppercased, *initial upper* if the initial character is uppercased, *contains digit* if it contains a digit and *other* if none of the previous rules applies. The capitalization feature is mapped to a seven-dimensional one-hot vector.

The word embedding, the capitalization feature and the character-based representation are concatenated (\parallel) and used for a BiLSTM-encoder. One LSTM network runs from the beginning of the sentence to the end while the other runs in reverse. The output of both LSTM networks are concatenated and are used as input for a classifier. For a Softmax classifier, we map the output through a dense layer with softmax as the activation function. This gives for each token in the sentence a probability distribution for the possible tags. The tag with the highest probability is selected. For the CRF classifier, the concatenated output is mapped with a dense layer and a linear activation function to the number of tags. Then, a linear-chain Conditional Random Field maximizes the tag probability of the complete sentence.

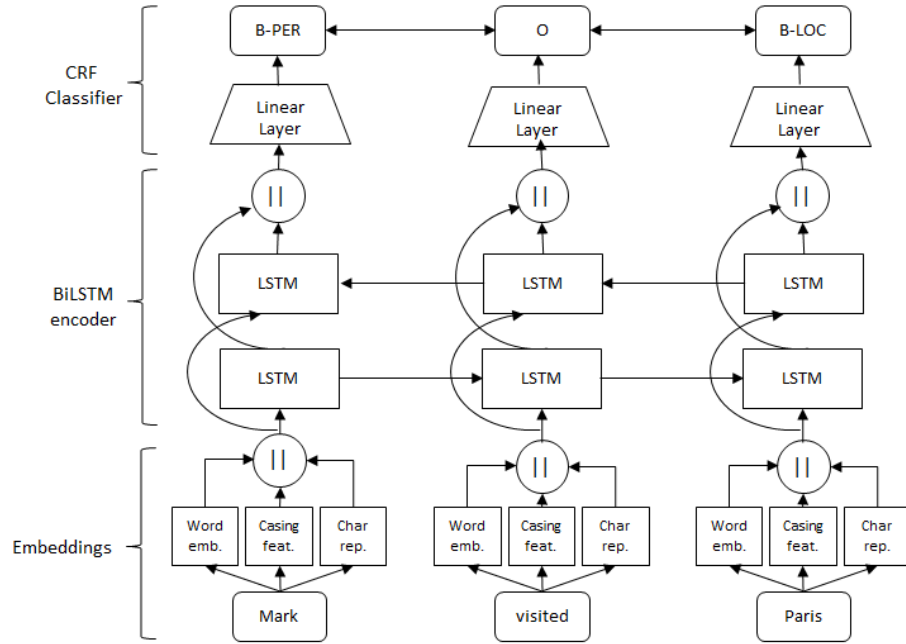


Figure 3.4: Our BiLSTM-CRF architecture used for sequence tagging.

3.2 Configurable Parameters of the BiLSTM-CRF Architecture

The BiLSTM-CRF architecture can be seen as a framework that has been proven useful for NLP sequence tagging task. However, the architectures published by [Huang et al. \(2015\)](#), [Ma and Hovy \(2016\)](#) and [Lample et al. \(2016\)](#) involve many different design choices and hyperparameters. It is unclear which parameters are relevant for the success of the architecture, hence, it is unclear on which parameters we should focus when tuning the architecture for a new task. Further, it is unclear if there exist default parameters that work well for a large number of tasks.

In this section, we summarize the configurable parameters of the BiLSTM-CRF architecture. In [section 3.5](#) we identify which parameters are relevant for tuning and, if possible, give default recommendations for the parameters.

Pre-trained Word Embeddings. Representing words as dense vectors, usually with 100 - 300 dimensions, is a widely used technique in NLP and it can significantly increase the performance (Collobert et al., 2011). Word embeddings provide a good generalization to unseen words since they can capture general syntactical as well as semantic properties of words. Which of the many published word embedding generation processes results in the best embeddings is unclear and depends on many factors, including the dataset from which they are created and for which purpose they will be used. For our benchmark datasets, we evaluate different popular, publicly available pre-trained word embeddings. We evaluate the **GoogleNews** embeddings¹ trained on part of the Google News dataset (about 100 billion words) from Mikolov et al. (2013), the Bag of Words (**Levy BoW**) as well as the dependency based embeddings (**Levy Dep.**)² by Levy and Goldberg (2014) trained on the English Wikipedia, three different **GloVe** embeddings³ from Pennington et al. (2014) trained either on Wikipedia 2014 + Gigaword 5 (about 6 billion tokens) or on Common Crawl (about 840 billion tokens), and the **Komninos and Manandhar (2016)** embeddings⁴ trained on the Wikipedia August 2015 dump (about 2 billion tokens). We also evaluate the approach of **FastText**, which does not train word embeddings directly, but trains embeddings for n-grams with length 3 to 6. The embedding of a word is then defined as the sum of the embeddings of the n-grams. This allows deriving a meaningful embedding even for rare words, which are often not part of the vocabulary for the other pre-trained embeddings. As the results in section 3.5.1 show, the different word embeddings lead to significant performance differences.

Character Representation. Character-level information, especially pre- and suffixes of words, can contain valuable information for linguistic sequence labeling tasks like part-of-speech tagging. However, instead of hand-engineered features, Ma and Hovy (2016) and Lample et al. (2016) present a method to learn task-specific character level representations while training. Ma and Hovy (2016) use convolutional neural networks (**CNNs**) (LeCun et al., 1989) to encode the trigrams of a word to a fixed-sized character-based representation. On the other hand, Lample et al. (2016) use bidirectional **LSTMs** to derive the character-based representation. In our experiments, we evaluate both approaches with the parameters mentioned in the respective papers.

Optimizer. The optimizer is responsible for the minimization of the objective function of the neural network. A commonly selected optimizer is stochastic gradient descent (**SGD**), which proved itself as an efficient and effective optimization method for a large number of published machine learning systems. However, SGD can be quite sensitive towards the selection of the learning rate. Choosing a too large rate can cause the system to diverge regarding the objective function, and choosing a too low rate results in a slow learning process. Further, SGD has troubles to navigate ravines and saddle points. To eliminate the shortcomings of SGD, other gradient-based optimization algorithms have been proposed. We evaluate the optimization

¹ <https://code.google.com/archive/p/word2vec/>

² <https://levyomer.wordpress.com/2014/04/25/dependency-based-word-embeddings/>

³ <http://nlp.stanford.edu/projects/glove/>

⁴ <https://www.cs.york.ac.uk/nlp/extvec/>

methods **Adagrad** (Duchi et al., 2011), **Adadelta** (Zeiler, 2012), **RMSProp** (Hinton, 2012), **Adam** (Kingma and Ba, 2014), and **Nadam** (Dozat, 2015), an Adam variant that incorporates Nesterov momentum (Nesterov, 1983). The results can be found in [section 3.5.3](#).

Gradient Clipping and Normalization. Two widely known issues with properly training recurrent neural networks are the *vanishing* and the *exploding* gradient problem (Bengio et al., 1994). While the vanishing gradient problem is countered by using LSTM networks, exploding gradients, i.e., gradients with extremely large values, are still an issue. Two common strategies to deal with the exploding gradient problem are **gradient clipping** (Mikolov, 2012) and **gradient normalization** (Pascanu et al., 2013). Gradient clipping involves clipping the gradient’s components element-wise if it exceeds a defined threshold τ . For each gradient component we compute $\hat{g}_{ij} = \max(-\tau, \min(\tau, g_{ij}))$. The matrix \hat{g} is then used for the weight update. Gradient normalization has a better theoretical justification and rescales the gradient whenever the norm $\|g\|_2$ goes over a threshold τ : $\hat{g} = (\tau/\|g\|_2)g$ if $\|g\|_2 > \tau$. In [section 3.5.4](#) we evaluate both approaches and their importance for a good test performance.

Tagging schemes. While the POS task assigns one label for each word in a sentence, many other sequence tagging tasks associate labels for segments in a sentence. This is achieved by using a special tagging scheme to identify the segment boundaries. We evaluate the **BIO**, **IOB**, and **IOBES** schemes. The **BIO** scheme marks the beginning of a segment with a B- tag and all other tokens of the same span with a I- tag. The O tag is used for tokens that are outside of a segment. The **IOB** scheme is similar to the BIO scheme, however, here the tag B- is only used to start a segment if the previous token is of the same class but is not part of the segment. The chunking data from CoNLL 2000 is provided using the BIO-scheme, while the NER dataset from CoNLL 2003 has an IOB tagging scheme. The **IOBES** scheme distinguishes between single token segments, which are tagged with an S- tag, the beginning of a segment that is tagged with a B- tag, the last token of a segment which is tagged with an E- tag, and tokens inside a segment which are tagged with an I- tag. It is unclear which of the three tagging scheme is in general better. Collobert et al. (2011) decided to use the most expressive IOBES tagging scheme for all their tasks. The results of our evaluations can be found in [section 3.5.5](#).

Note, when a tagging scheme is used, the classifier might produce invalid tags, for example, an I- tag without a previous B- tag to start the segment. We observed a high number of invalid tags especially for the softmax classifier which does not take the label sequence into account. For the CRF classifier, invalid tags occurred only rarely. Depending on the used evaluation script such invalid tags might result in an erroneous computation of the performance score. To eliminate invalid tags, we applied two post-processing strategies: Either set all invalid tags to O or change the tag to B- to start a new segment. Usually setting all invalid tags to O resulted in slightly superior results.

Classifier. We evaluated two options for the last layer of the network. The first option is a dense layer with softmax activation function, i.e., a **softmax classifier**. This classifier produces a probability distribution for the different tags for each token

in the sentence. In this approach, each token in a sentence is considered independently and correlations between tags in a sentence cannot be taken into account. The second option we evaluate is a dense layer with a linear activation function followed by a linear-chain Conditional Random Field (CRF). We call this variant **CRF classifier**. This option is able to maximize the tag probability of the complete sentence. This is especially helpful for tasks with strong dependencies between token tags, for example, if one of the described tagging schemes is used where certain tags cannot follow other tags. This approach is also known as BiLSTM-CRF (Huang et al., 2015). The results can be found in [section 3.5.6](#).

Dropout. Dropout is a popular method to deal with overfitting for neural networks (Srivastava et al., 2014a). In our setup, we evaluate three options: **No dropout**, **naive dropout**, and **variational dropout**. *Naive dropout* is the simplest form of dropout: We apply a randomly selected dropout mask for each LSTM-output. The mask changes from time step to time step. The recurrent connections are not dropped. As noted by Gal and Ghahramani (2016), this form of dropout is suboptimal for recurrent neural networks. They propose to use the same dropout mask for all time steps of the recurrent layer, i.e., at each time step the same positions of the output are dropped. They also propose to use the same strategy to drop the recurrent connections. This approach is known as *variational dropout*. Further details can be found in (Gal and Ghahramani, 2016). The fraction p of dropped dimensions is a hyperparameter and is selected randomly from the set $\{0.0, 0.05, 0.1, 0.25, 0.5\}$. Note, for variational dropout, the fraction p is selected independently for the output units as well as for the recurrent units. The results can be found in [section 3.5.7](#).

Number of LSTM-Layers. We evaluated 1, 2, and 3 stacked BiLSTM-layers. The results can be found in [section 3.5.8](#).

Number of Recurrent Units. The number of recurrent units was selected from the set $\{25, 50, 75, 100, 125\}$. The forward and reverse running LSTM-networks had the same number of recurrent units. In the case of multiple layers, we selected for each BiLSTM-layer a new value. Increasing layers sizes were forbidden. For [section 3.5.8](#), we evaluated networks with $60 \leq u \leq 300$ recurrent units.

Mini-batch Size. We evaluated mini-batch sizes of 1, 8, 16, 32, and 64 sentences.

3.3 Benchmark Datasets

We will show in [chapter 5](#), that it is advisable to evaluate approaches on multiple (similar) tasks, especially if we are interested to find universal machine learning approaches. Hence, we selected some common NLP sequence tagging tasks as well as one event detection task to evaluate the described BiLSTM-CRF architecture.

Part-of-Speech tagging. Part-of-Speech tagging aims at labeling each token in the text with a tag indicating its syntactic role, e.g., noun, verb etc. The typical benchmark setup is described in detail in (Toutanova et al., 2003). Usually, the

sections 0-18 of the Wall Street Journal (WSJ) are used for training, while sections 19-21 are used for validation and hyperparameter optimization, and sections 22-24 are used for testing.

Part-of-Speech tagging is a relatively simple task. Toutanova et al. (2003) report an accuracy of 97.24% on the test set. The estimated error rate for the PennTree Bank POS information is approximately 3% (Marcus et al., 1993). Marcus et al. (1993) found in an early experiment on the Brown corpus that the disagreement between two annotators when correcting the output of an automatic tagger is 4.1% and 3.5% once one difficult text is excluded. An improvement lower than the error rate, i.e., above 97% accuracy, is unlikely to be meaningful and has a high risk of being the result of chance.

To be able to compare different neural architectures and hyperparameters for Part-of-Speech tagging, we increased the difficulty of the task by decreasing the training set. We decided to decrease the size of the training set to the first 500 sentences in the Wall Street Journal. The development and test sets were kept unchanged. This decreased the accuracy to a range of about 94-95%, fairly below the (estimated) upper-bound of 97%.

Chunking. Chunking aims at labeling segments of a sentence with syntactic constituents such as noun or verb phrase. Each word is assigned a single tag. To note the beginning of a new segment, a tagging scheme, for example, a BIO tagging scheme, can be used. Here, the tag B-NP denotes the beginning of a noun phrase and I-NP would denote each other word of the noun phrase. We evaluate our systems using the CoNLL 2000 shared task⁵. Sections 15-18 of the Wall Street Journal are used for training, section 19 is used as development set, and section 20 is used for testing. The performance is computed using the F_1 score, which is the harmonic mean of the precision and recall. Note that besides the class label the span of the segment must perfectly match the gold data. If a produced segment is a single token too long or too short, it is considered an error.

NER. Named Entity Recognition (NER) aims at labeling named entities, like person names, locations, or company names, in a sentence. As in the chunking task, a named entity can consist of several tokens, and a tagging scheme is involved to denote the beginning and the end of an entity. We use the CoNLL 2003 setup⁶ which provides train, development and test data.

Entities. In contrast to the CoNLL 2003 NER dataset, the ACE 2005 dataset⁷ annotated not only named entities, but all words referring to an entity, for example the words *U.S. president*. The entities are categorized in the seven categories: *persons*, *organizations*, *geographical/social/political entities*, *locations*, *facilities*, *vehicle* and *weapon*. We use the same data split as Li et al. (2013) partitioning the data into 529 train documents, 40 development documents, and 30 test documents.

Events. We use the TempEval3 Task B⁸ (UzZaman et al., 2013) dataset that is

⁵ <http://www.cnts.ua.ac.be/conll2000/chunking/>

⁶ <http://www.cnts.ua.ac.be/conll2003/ner/>

⁷ <https://catalog.ldc.upenn.edu/LDC2006T06>

⁸ <https://www.cs.york.ac.uk/semeval-2013/task1/>

annotated with the TimeML scheme. The dataset is discussed in detail in [chapter 2](#).

An overview of the datasets is given in [Table 3.1](#). We observe that the number of train, development and test sentences varies significantly between the datasets. Further, the relative size of the development and test set varies strongly between the datasets. The split from [Li et al. \(2013\)](#) for the ACE 2005 dataset uses only about 4% of the annotated sentences for the test set, while it is about 16% for the GermEval 2014 dataset.

Task	Train	Dev	Test	#tags
Penn TreeBank - POS	500	5524	5459	45
ACE 2005 - Entities	15185	882	674	15
CoNLL 2000 - Chunking	8926	1844	2009	23
CoNLL 2003 - NER	13862	3250	3420	9
TempEval-3 - Events	4090	149	279	3

Table 3.1: Overview of the benchmark tasks and number of sentences for the training, development and test set.

3.4 Evaluation Methodology

We try to identify design choices that improve **robustly** the performance of the BiLSTM-CRF architecture, i.e., design choices and parameter selections that improve the performance independent of the remaining configuration of the architecture. For example, we want to study if a CRF classifier can lead to an improvement in comparison to a softmax classifier independent of the remaining configuration of the network.

In order to find the most robust design options, we sampled randomly a large number of network configurations and evaluated each configuration with each variation. For example, to decide whether a softmax classifier or a CRF classifier as the last layer is better, we sampled for each task hundreds of network configurations and evaluated whether the version with a softmax classifier or with a CRF classifier performed better.

The achieved test performances can be plotted as a violin plot ([Figure 3.5](#)). The violin plot is similar to a boxplot, however, it estimates from the samples the probability density function and depicts it along the Y-axis. If a violin plot is wide at a certain location, then achieving this test performance is especially likely. Besides the probability density, it also shows the median as well as the quartiles. In [Figure 3.5](#) we can observe that a CRF classifier usually results in a higher performance than a softmax classifier for the chunking dataset. As we sample and run several hundred hyperparameter configurations, random noise, e.g. from the weight initialization, will cancel out and we can conclude that the CRF classifier is a better option for this task.

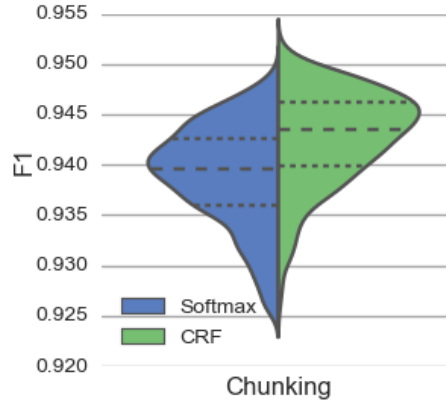


Figure 3.5: Probability density function for the chunking task using a softmax classifier or a CRF classifier as a last layer of a BiLSTM-network. The median and the quartiles are plotted as dashed lines.

For brevity reasons, we show the violin plot only in certain situations. In most cases, we report a table like [Table 3.2](#). The table shows how many network configurations were sampled randomly for each task with the parameters described in [section 3.2](#). For the Chunking task, 230 configurations were sampled randomly. Each configuration was evaluated with a softmax classifier as well as a CRF classifier as the last layer of the network. For 219 of the 230 configurations (95.2%), the CRF setup achieved a better test performance than the setup with a softmax classifier. The average depicted at the bottom of the table is the macro average about how often each evaluated option achieved the best performance for the five tasks.

Besides measuring which option achieves the better performance, we also compute the median of the differences to the best option. For the Chunking task, the option that resulted in most cases in the best performance was the CRF classifier. Hence, for the softmax option, we compute the median difference of test performance to the best option. Let S_i be the test performance (F_1 -measure) for the softmax setup for configuration i and C_i the test performance for the CRF setup. We then compute $\Delta F_1 = \text{median}(S_1 - C_1, S_2 - C_2, \dots, S_{230} - C_{230})$. For the chunking task, the median difference was $\Delta F_1 = -0.38\%$, i.e., the setup with a softmax classifier achieved on average an F_1 -score of 0.38 percentage points below that of the CRF setup.

Based on the outcome of the different runs, we use a two-sided binomial test to find options that perform *statistically significant* better than other options. As threshold we use $p < 0.01$. We compute the standard deviation σ of the achieved performance scores to measure the dependence on the remaining configuration of the network and on the random seed value. We use a Brown-Forsythe test with threshold $p < 0.01$ to identify standard deviations that are statistically significant from others. The best result and all statistically equal results are marked with a †. If none of the options in a row has a †, then we did not observe a statistically significant difference between the options.

Task	# Configs	Softmax	CRF
POS	114	19.3%	80.7% †
$\Delta Acc.$		-0.19%	
σ		0.0149	0.0132
Chunking	230	4.8%	95.2% †
ΔF_1		-0.38%	
σ		0.0058	0.0051
NER	235	9.4%	90.6% †
ΔF_1		-0.67%	
σ		0.0081	0.0060†
Entities	214	13.1%	86.9% †
ΔF_1		-0.85%	
σ		0.0157	0.0140
Events	203	61.6% †	38.4%
ΔF_1			-0.15%
σ		0.0052	0.0057
Average		21.6%	78.4%

Table 3.2: Network configurations were sampled randomly and each was evaluated with each classifier as a last layer. The first number in a cell depicts in how many cases each classifier produced better results than the others. The second number shows the median difference to the best option for each task. Statistically significant differences with $p < 0.01$ are marked with †.

Limitations

Our evaluation methodology studies the parameters and design choices independently of each other. However, this has the limitation that dependencies between parameters are not evaluated well. For example, it might be that a network with more recurrent nodes requires a higher dropout value to work well. Further, the identified parameter values that work best in isolation must not be the best overall configuration of the architecture. Taking dependencies between parameters into account might yield a better default configuration for new tasks.

Nonetheless, identifying design choices and parameters that work well independent of the remaining configuration is hugely valuable for applying this architecture to new tasks. For a new task, we usually cannot ensure that every aspect of the architecture is optimally set. Hence, to serve as a uniform learning approach, the approach should work well even when some parts are configured sub-optimal.

3.5 Evaluation Results

The following table gives an overview of the results of our experiments. Details can be found in the consecutive subsections.

Parameter	Recom. Config	Impact	Comment
Word Embeddings	Komninos et al.	High	The embeddings by Komninos and Manandhar (2016) resulted for the most tasks in the best performance. For the POS tagging task, the median difference to e.g. the GloVe embeddings was 4.97 percentage points. The GloVe embeddings trained on Common Crawl were especially well suited for the NER task, due to their high coverage. More details in section 3.5.1 .
Character Representation	CNNs (Ma and Hovy, 2016)	Low - Medium	Character-based representations were in a lot of tested configurations not that helpful and could not improve the performance of the network. The CNN approach by Ma and Hovy (2016) and the LSTM approach by Lample et al. (2016) performed on-par. The CNN approach should be preferred due to the higher computational efficiency. More details in section 3.5.2 .
Optimizer	Nadam	High	Adam and Adam with Nesterov momentum (Nadam) usually performed the best, followed by RMSProp. Adadelata and Adagrad had a much higher variance regarding test performance and resulted on average to far worse results. SGD failed in a high number of cases to converge to a minimum, likely due to its high sensitivity of the learning rate. Nadam was the fastest optimizer. More details in section 3.5.3 .
Gradient Clipping / Normalization	Gradient Normalization with $\tau = 1$	High	Gradient clipping does not improve the performance. Gradient normalization as described by Pascanu et al. (2013) improves significantly the performance with an observed average improvement between 0.45 and 0.82 percentage points. The threshold τ is of minor importance, with $\tau = 1$ giving usually the best results. More details in section 3.5.4 .

Tagging Scheme	BIO	Medium	The BIO and IOBES tagging scheme performed consistently better than the IOB tagging scheme. IOBES does not give a significant performance increase compared to the BIO tagging scheme. More details in section 3.5.5 .
Classifier	CRF	High	Using a CRF instead of a softmax classifier as the last layer gave a large performance increase for tasks with a high dependency between tags. This was also true for stacked BiLSTM layers. For tasks without dependencies between the tags, softmax performed better. More details in section 3.5.6 .
Dropout	Variational	High	Variational dropout (Gal and Ghahramani, 2016) outperformed significantly no dropout and also naive dropout. The best result was achieved when dropout was applied both to the output units as well as to the recurrent units of the LSTM networks. More details in section 3.5.7 .
#LSTM Layers	2	Medium	If the number of recurrent units is kept constant, two stacked BiLSTM-layers resulted in the best performance. More details in section 3.5.8 .
Recurrent Units	100	Low	The number of recurrent units, as long as it is not far too large or far too small, has only a minor effect on the results. A value of about 100 for each LSTM-network appears to be a good rule of thumb for the tested tasks. More details in section 3.5.9 .
Mini-batch Size	1-32	Medium	The optimal size for the mini-batch appears to depend on the task. For POS tagging and event recognition, a size of 1 was optimal, for chunking a size of 8, and for NER and Entity Recognition a size of 32. More details in section 3.5.10 .

3.5.1 Word Embeddings

Table 3.4 shows the impact of different pre-trained word embeddings on the five evaluated benchmark tasks. The embeddings by Komninos and Manandhar (2016) give the best performance on all tasks except for the CoNLL 2003 NER and CoNLL 2000 chunking tasks, where they perform on-par with the GloVe embeddings on CommonCrawl and the Levy and Goldberg (2014) dependency-based embeddings. The median difference in test performance is quite large. For example, for the POS tagging task, the embeddings by Komninos and Manandhar (2016) give on average a 4.97 percentage points higher accuracy than the GloVe2 embeddings (100 dimensions, trained on Wikipedia and Gigaword).

The only datasets where the Komninos and Manandhar (2016) embeddings would not necessarily be the best selection is for Chunking and the NER task. For Chunking, the dependency based embeddings by Levy and Goldberg (2014) gave in most cases the best performance. However, this difference is statistically not significant ($p=6.5\%$) and looking at the mean performance shows that both embeddings are on-par.

For the NER task, the GloVe embeddings trained on 840 billion tokens from Common Crawl (GloVe 3) resulted in most cases in the best performance. As before, the difference to the Komninos and Manandhar (2016) embeddings is statistically insignificant ($p=33.0\%$) and more runs would be required to determine which embeddings would be the best selection.

The n-gram embedding approach FastText by Bojanowski et al. (2016), which allows deriving meaningful word embeddings also for rare words which are often not in the vocabulary for the other approaches, does not yield a good performance in any of the benchmark tasks.

Conclusion. The selection of the pre-trained word embeddings has a large impact on the performance of the system, a much larger impact than many other hyperparameters. In most tasks, the Komninos and Manandhar (2016) embeddings gave by a far margin the best performance. Pre-trained word embeddings provide valuable information about syntactic and semantic relationships between words. This is especially crucial for words that do not appear in the training corpus or only occur infrequently. Embeddings based on dependency relations better capture functional properties of words, while embeddings based on context windows better capture topical similarity between words (Komninos and Manandhar, 2016). Komninos and Manandhar combined the idea of dependency-based embeddings with a context-based approach. It appears that this gives a good representation of the functional and of the semantic properties of words.

Dataset	L-Dep.	L-BoW	GloVe1	GloVe2	GloVe3	Komn.	G.News	Fast
POS	6.5%	0.0%	0.0%	0.0%	0.0%	93.5% †	0.0%	0.0%
$\Delta Acc.$	-0.39%	-2.52%	-4.14%	-4.97%	-2.60%		-1.95%	-2.28%
σ	0.0125†	0.0147	0.0203	0.0136	0.0097	0.0058†	0.0118	0.0120
Chunk.	60.8% †	0.0%	0.0%	0.0%	0.0%	37.1%†	2.1%	0.0%
ΔF_1		-0.52%	-1.09%	-1.50%	-0.93%	-0.10%	-0.48%	-0.75%
σ	0.0056	0.0065	0.0094	0.0083	0.0070	0.0044†	0.0064	0.0068
NER	4.5%	0.0%	22.7%†	0.0%	43.6% †	27.3%†	1.8%	0.0%
ΔF_1	-0.85%	-1.17%	-0.15%	-0.73%		-0.08%	-0.75%	-0.89%
σ	0.0073†	0.0084†	0.0077†	0.0081	0.0069†	0.0064†	0.0081†	0.0075†
Entities	4.2%	7.6%	0.8%	0.0%	6.7%	57.1% †	21.8%	1.7%
ΔF_1	-0.92%	-0.89%	-1.50%	-2.24%	-0.80%		-0.33%	-1.13%
σ	0.0167	0.0170	0.0178	0.0180	0.0154	0.0148	0.0151	0.0166
Events	12.9%	4.8%	0.0%	0.0%	0.0%	71.8% †	9.7%	0.8%
ΔF_1	-0.55%	-0.78%	-2.77%	-3.55%	-2.55%		-0.67%	-1.36%
σ	0.0045†	0.0049†	0.0098	0.0089	0.0086	0.0060	0.0066	0.0062
Average	17.8%	2.5%	4.7%	0.0%	10.1%	57.4%	7.1%	0.5%

Table 3.4: Randomly sampled configurations were evaluated with different pre-trained word embeddings. The first number depicts for how many configurations each setting resulted in the best test performance. The second number shows the median difference to the best option for each task. 108 configurations were sampled for POS, 97 for Chunking, 110 for NER, 119 for Entities, and 124 for Events. Statistically significant differences with $p < 0.01$ are marked with †. *L-Dep.* dependency based embeddings by [Levy and Goldberg \(2014\)](#), *L-BoW* Bag-of-Word embeddings by Levy and Goldberg, *GloVe1*: Wikipedia 2014 + Gigaword 5, 100 dimensions, *GloVe2*: Wikipedia 2014 + Gigaword 5, 300 dimensions, *GloVe3*: Common Crawl embeddings, *Komn.* embeddings by [Komninos and Manandhar \(2016\)](#), *G.News* [Mikolov et al. \(2013\)](#) embeddings on Google News Corpus, *Fast* embeddings by [Bojanowski et al. \(2016\)](#).

3.5.2 Character Representation

We evaluate the approaches of [Ma and Hovy \(2016\)](#) using Convolutional Neural Networks (CNN) as well as the approach of [Lample et al. \(2016\)](#) using LSTM-networks to derive character-based representations.

[Table 3.5](#) shows that character-based representations yield a statistically significant difference only for the POS, the Chunking, and the Events task. For NER and Entities, the difference to not using a character-based representation is not significant ($p > 0.01$).

The difference between the CNN approach by [Ma and Hovy \(2016\)](#) and the LSTM approach by [Lample et al. \(2016\)](#) to derive character-based representations is statistically insignificant. This is quite surprising, as both approaches have fundamentally different properties: The CNN approach from [Ma and Hovy \(2016\)](#) takes only trigrams into account. It is also position independent, i.e., the network will not be able to distinguish between trigrams at the beginning, in the middle, or at the end of a word, which can be crucial information for some tasks. The BiLSTM approach from [Lample et al. \(2016\)](#) takes all characters of the word into account. Further, it

is position-aware, i.e., it can distinguish between characters at the start and at the end of the word. Intuitively, one would think that the LSTM approach by Lample et al. would be superior.

Task	# Configs	No	CNN	LSTM
POS	225	4.9%	58.2% †	36.9%
$\Delta Acc.$		-0.90%		-0.05%
σ		0.0201	0.0127†	0.0142†
Chunking	241	13.3%	43.2%†	43.6% †
ΔF_1		-0.20%	-0.00%	
σ		0.0084	0.0067†	0.0065†
NER	217	27.2%	36.4%	36.4%
ΔF_1		-0.11%		-0.01%
σ		0.0082	0.0082	0.0080
Entities	228	26.8%	36.0%	37.3%
ΔF_1		-0.07%	0.00%	
σ		0.0177	0.0165	0.0171
Events	219	20.5%	35.6%†	43.8% †
ΔF_1		-0.44%	-0.04%	
σ		0.0140	0.0103†	0.0096†
Average		18.5%	41.9%	39.6%

Table 3.5: Comparison of not using character-based representations and using CNNs (Ma and Hovy, 2016) or LSTMs (Lample et al., 2016) to derive character-based representations. The first number depicts for how many configurations each setting resulted in the best test performance. The second number shows the median difference to the best option for each task. Statistically significant differences with $p < 0.01$ are marked with †.

Note that we tested both options only with the presented hyperparameters in their respective papers. Each character was mapped to a randomly initialized 30-dimensional embedding. For the CNN approach, Ma and Hovy (2016) used 30 filters and a filter length of 3, which yields a 30-dimensional representation for each word. For the bidirectional LSTM approach, Lample et al. (2016) used 25 recurrent units, yielding a character-based representation of 50 dimensions for each word. It would be of interest if the performance could be improved by selecting different hyperparameters, for example for the CNN approach to not only use character trigrams but also using shorter and/or longer n-grams.

Conclusion. Character-based representations were in a lot of the tested configurations not that helpful and could not improve the performance of the network. The CNN approach by Ma and Hovy (2016) and the LSTM approach by Lample et al. (2016) performed on-par. The CNN approach should be preferred due to the higher computational efficiency. For tasks that focus on semantic properties of words (i.e. the NER and Entities tasks), there was no significant improvement by using character-based representations.

3.5.3 Optimizers

We evaluated different optimizers for the network: Stochastic Gradient Descent (*SGD*), *Adagrad* (Duchi et al., 2011), *Adadelta* (Zeiler, 2012), *RMSProp* (Hinton, 2012), *Adam* (Kingma and Ba, 2014), and *Nadam* (Dozat, 2015), an Adam variant that incorporates Nesterov momentum (Nesterov, 1983). For SGD, we tuned the learning rate by hand, however, we could observe that SGD failed in many instances to converge to a minimum. For the other optimizers, we used the recommended settings from the respective papers.

Figure 3.6 and Figure 3.7 show the performance for the different choices of optimizers for the Chunking and the NER task, respectively. Table 3.6 contains the results for all other tasks.

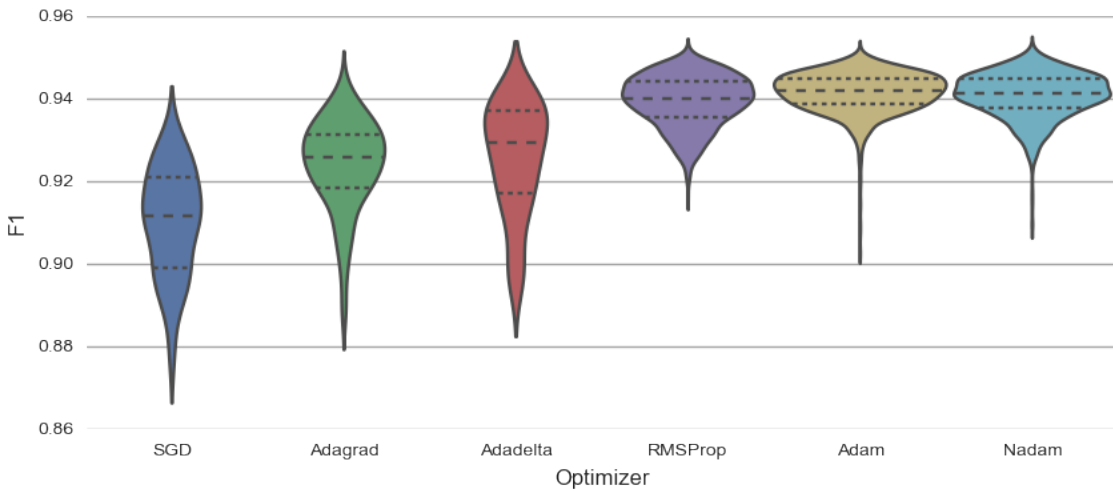


Figure 3.6: Performance on the CoNLL 2000 chunking shared task for various optimizers.

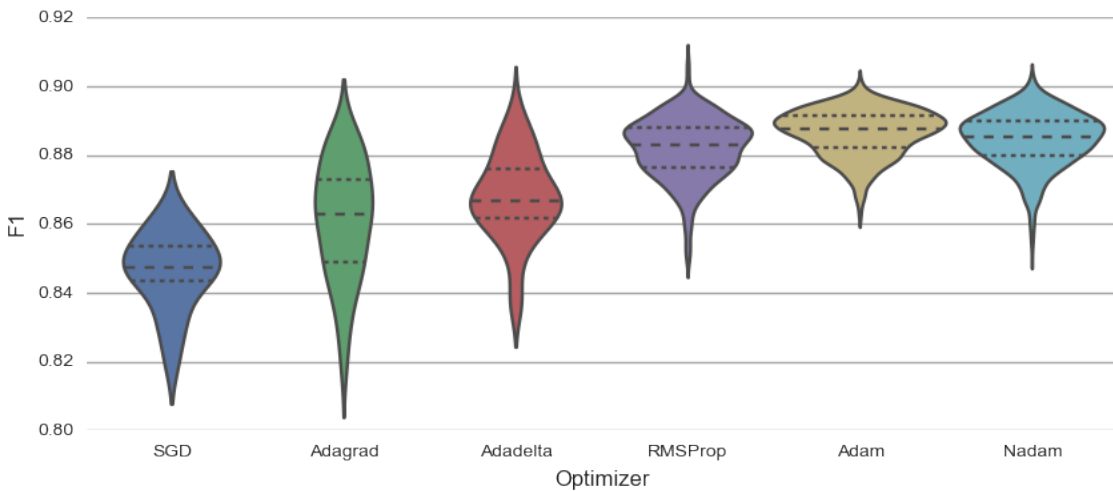


Figure 3.7: Performance on the CoNLL 2003 NER shared task for various optimizers.

Task	Configs	Adam	Nadam	RMSProp	Adadelta	Adagrad	SGD
POS	218	7.3%	82.6% †	10.1%	0.0%	0.0%	0.0%
$\Delta Acc.$		-0.33%		-0.30%	-4.08%	-1.93%	-18.08%
σ		0.0210†	0.0172†	0.0200†	0.1915	0.0463	0.2685
Median epochs		31	20	31	47	40	46
Chunking	192	17.2%	51.6% †	31.2%†	0.0%	0.0%	0.0%
ΔF_1		-0.11%		-0.04%	-0.90%	-1.23%	-3.74%
σ		0.0090†	0.0085†	0.0085†	0.0125	0.0135	0.2958
Median epochs		16	10	14	37	31	39
NER	207	25.1%†	36.7% †	34.8%†	1.9%	1.4%	0.0%
ΔF_1		-0.10%		0.03%	-0.77%	-0.82%	-5.38%
σ		0.0092†	0.0091†	0.0096†	0.0138	0.0139	0.1328
Median epochs		12	9	10	22	19	42
Entities	152	25.0%†	40.8% †	30.3%†	3.9%	0.0%	0.0%
ΔF_1		-0.14%		-0.09%	-1.49%	-1.82%	-6.00%
σ		0.0167†	0.0166†	0.0165†	0.0244	0.0288	0.1960
Median epochs		13	10	11	32	29	46
Events	113	17.7%†	31.9% †	26.5%†	8.0%	15.0%	0.9%
ΔF_1		-0.10%		-0.05%	-0.55%	-0.34%	-1.68%
σ		0.0129†	0.0127†	0.0142†	0.0142†	0.0155†	0.0644
Median epochs		8	5	7	12	7	19
Average		18.5%	48.7%	26.6%	2.8%	3.3%	0.2%

Table 3.6: Randomly sampled network configurations were evaluated with each optimizer. The first number depicts in how many cases each optimizer produced better results than the others. The second number shows the median difference to the best option for each task. Median epochs depict the median number of train epochs until convergence. Statistically significant differences with $p < 0.01$ are marked with †.

We observe that the variance for RMSProp, Adam, and Nadam is much smaller in comparison to SGD, Adagrad, and Adadelta. Here, we can conclude that these optimizers are able to achieve a better performance independent of the remaining configuration of the network and/or of the random seed value.

Nadam showed the best performance, yielding the highest score for 48.7% of the tested configurations. The difference between Nadam and Adam or RMSProp is often not statistically significant, i.e., more experiments would be required to determine which optimizer yields the best performance.

We measured the time an optimizer required to converge. The time per epoch was for all optimizers similar. However, we observed a large difference in the number of epochs until convergence. Nadam converged the fastest, i.e., requiring only a small number of training epochs to achieve a good performance. Adadelta, Adagrad, and SGD required the longest to converge to a minimum.

Kingma and Ba (2014) recommend for the Adam optimizer certain default parameters. However, we did some manual tuning for our tasks and increased the learning rate for Adam to 0.01 for the first three epochs, set it to 0.005 for the consecutive three epochs and then setting it back to its default value of 0.001. The result is depicted in Table 3.7. Not only does this lead to faster convergence, but it also leads to better performance for the POS tagging and the Chunking task. It would

be of interest to evaluate the other hyperparameters of Adam and Nadam to see their impact on the performance.

Task	Configs	Adam	Adam (increased LR)
POS $\Delta Acc.$ Median train epochs	226	35.0% (-0.15%) (31)	65.0% (22)
Chunking ΔF_1 Median train epochs	203	31.0% (-0.17%) (15)	69.0% (7)
NER ΔF_1 Median train epochs	224	46.9% (-0.08%) (12)	53.1% (7)
Entities ΔF_1 Median train epochs	197	56.3% (16)	43.7% (-0.15%) (9)
Events ΔF_1 Median train epochs	210	50.5% (8)	49.5% (-0.00%) (5)
Average		43.9%	56.1%

Table 3.7: Comparison of Adam to Adam with increase learning rate (LR). The first number depicts in how many cases each optimizer produced better results than the others. The second number shows the median difference to the best option for each task. Median epochs depict the median number of train epochs until convergence.

Conclusion. Adam, Nadam, and RMSProp produced more stable and better results than SGD, Adagrad, or Adadelta. In our experiments, Nadam (Adam with Nesterov momentum) was on average the best optimizer. RMSProp produced test scores on average up to 0.30 percentage points below of Adam or Nadam. Nadam had the best convergence time, i.e., required the lowest number of epochs. Adapting the learning rate for Adam can further improve the performance as well as the convergence time. The results in our experiments show that more recent optimizers find more reliably good minima with fewer trainings epoch.

3.5.4 Gradient Clipping and Normalization

Two common strategies to deal with the exploding gradient problem are *gradient clipping* (Mikolov, 2012) and *gradient normalization* (Pascanu et al., 2013). Gradient clipping involves clipping the gradient’s components element-wise if they exceed a defined threshold. Gradient normalization has a better theoretical justification and rescales the gradient whenever the norm goes over a threshold.

The results for gradient clipping are depicted in Table 3.8. For the evaluated thresholds, we could not observe any statistically significant improvements for the five tasks.

Gradient normalization has a better theoretical justification (Pascanu et al., 2013), and we observe that it leads to better performance as depicted in Table 3.9. The concrete threshold value for the gradient normalization is of lower importance, as long as it is not too small or too large. A threshold value of 1 performed the best in most cases.

Task	# Configs	Clipping threshold				
		None	1	3	5	10
POS	106	24.5%	21.7%	20.8%	17.9%	15.1%
$\Delta Acc.$			-0.00%	-0.02%	-0.02%	-0.02%
σ		0.2563	0.2649	0.2407	0.2809	0.2715
Chunking	109	24.8%	26.6%	15.6%	13.8%	19.3%
ΔF_1		-0.05%		-0.02%	-0.03%	-0.03%
σ		0.1068	0.0101	0.0765	0.1160	0.0111
NER	84	16.7%	25.0%	22.6%	17.9%	17.9%
ΔF_1		-0.04%		-0.00%	-0.03%	0.02%
σ		0.0110	0.0110	0.0104	0.0111	0.0114
Entities	85	18.8%	16.5%	21.2%	22.4%	21.2%
ΔF_1		-0.10%	-0.07%	-0.04%		-0.07%
σ		0.0176	0.0167	0.0188	0.0190	0.0203
Events	99	21.2%	17.2%	16.2%	28.3%	17.2%
ΔF_1		-0.02%	-0.14%	-0.01%		-0.05%
σ		0.0156	0.0161	0.0167	0.0158	0.0153
Average		21.2%	21.4%	19.3%	20.0%	18.1%

Table 3.8: Element-wise clipping of gradient values to a certain threshold, as described by Mikolov (2012). The tested thresholds 1, 3, 5 and 10 did not lead to any improvement compared to not clipping the gradient. Statistically significant differences with $p < 0.01$ are marked with †.

Conclusion. Gradient clipping does not improve the performance. Gradient normalization as described by Pascanu et al. (2013) significantly improves the performance with an observed average improvement between 0.45 and 0.82 percentage points. The threshold τ is of minor importance, with $\tau = 1$ usually giving the best results.

Task	# Configs	Normalization threshold				
		None	1	3	5	10
POS	106	3.8%	40.6% †	24.5%†	15.1%†	16.0%
$\Delta Acc.$		-0.82%		-0.05%	-0.05%	-0.08%
σ		0.2563	0.0254†	0.0245†	0.0253†	0.0254†
Chunking	109	6.4%	27.5% †	24.8%†	22.0%†	19.3%†
ΔF_1		-0.29%		-0.00%	-0.04%	-0.04%
σ		0.1068	0.0087	0.0087	0.0088	0.0086
NER	84	7.1%	32.1% †	20.2%†	23.8%†	16.7%†
ΔF_1		-0.44%		-0.05%	-0.10%	-0.14%
σ		0.0110	0.0101	0.0101	0.0100	0.0112
Entities	87	6.9%	21.8%†	23.0%†	27.6% †	20.7%†
ΔF_1		-0.58%	-0.02%	0.09%		-0.01%
σ		0.0831	0.0168	0.0158	0.0172	0.0163
Events	106	3.8%	30.2% †	26.4%†	21.7%†	17.9%†
ΔF_1		-0.59%		-0.03%	-0.09%	-0.21%
σ		0.0157	0.0143	0.0137	0.0147	0.0141
Average		5.6%	30.5%	23.8%	22.0%	18.1%

Table 3.9: Normalizing the gradient to $\hat{g} = (\tau/\|g\|_2)g$ if the norm $\|g\|_2$ exceeds a threshold τ (Pascanu et al., 2013). We see a clear performance increase in comparison to not normalizing the gradient. The threshold value τ is of minor importance, with $\tau = 1$ usually performing the best. Statistically significant differences with $p < 0.01$ are marked with †.

3.5.5 Tagging Schemes

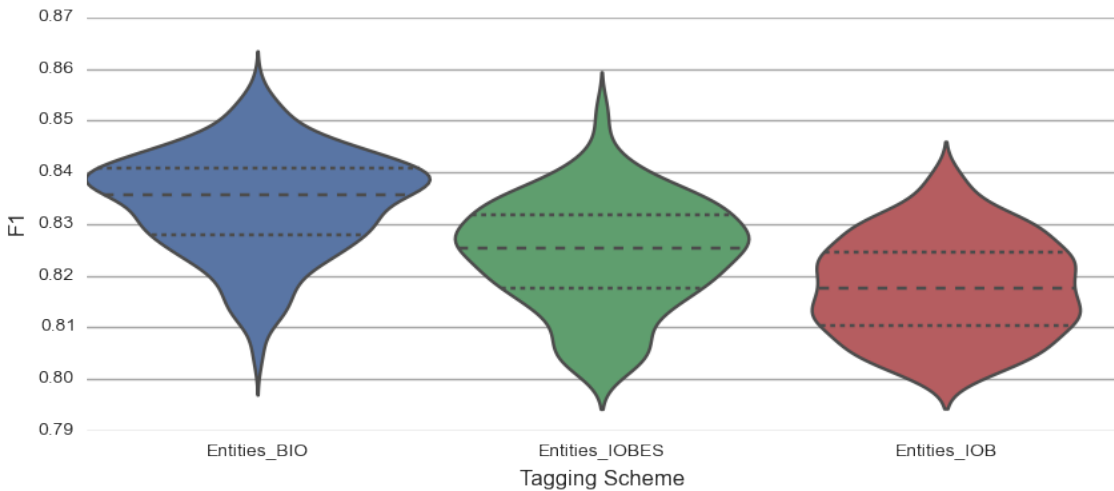


Figure 3.8: Performance on the ACE 2005 entities dataset for various tagging schemes.

Figure 3.8 depicts the violin plot for different tagging schemes for the ACE 2005 entities recognition task. Table 3.10 summarizes the results for all other tasks. The

IOB tagging scheme performs poorly on all tasks. The BIO and IOBES tagging schemes perform on-par, except for the Entities dataset. There, we observe a much better performance for the BIO scheme.

Task	# Configs	BIO	IOB	IOBES
Chunking	106	38.7% [†]	4.7%	56.6%[†]
ΔF_1		-0.07%	-0.34%	
σ		0.0052	0.0061	0.0048
NER	98	40.8% [†]	7.1%	52.0%[†]
ΔF_1		-0.09%	-0.46%	
σ		0.0074	0.0084	0.0066
Entities	106	88.7%[†]	0.0%	11.3%
ΔF_1			-1.90%	-1.01%
σ		0.0108 [†]	0.0162	0.0142 [†]
Events	107	47.7%[†]	9.3%	43.0% [†]
ΔF_1			-0.34%	-0.05%
σ		0.0038	0.0039	0.0044
Average		54.0%	5.3%	40.7%

Table 3.10: Network configurations were sampled randomly and were evaluated with each tagging scheme. The first number in the cell depicts in how many cases each tagging scheme produced better results than the others. The second number shows the median difference to the best option for each task. Statistically significant differences with $p < 0.01$ are marked with [†].

Conclusion. The IOB tagging scheme produced by far the worst results, which can be due to the rather unintuitive definition of it. The B-tag is only used to separate two directly consecutive tags of the same type. Otherwise, the I-tag is used to start new tags or to continue a tag. The BIO and IOBES tagging schemes were producing similar results, i.e., the more expressive IOBES tagging scheme did not yield a significant improvement. We would recommend using the BIO tagging scheme due to the lower overhead.

3.5.6 Classifier - Softmax vs. CRF

Figure 3.9 depicts the difference between a softmax classifier as final layer versus optimizing the complete label sequence for the whole sentence using a Conditional Random Field (CRF). The violin plots show a clear preference for a CRF classifier as the final layer for all except the TempEval-3 events dataset. Table 3.11 compares the two options when all other hyperparameters are kept the same. It confirms the impression that CRF leads to superior results in most cases, except for the event detection task. The improvement by using a CRF classifier instead of a softmax classifier lies between 0.19 percentage points and 0.85 percentage points for the evaluated tasks.

When using the BIO- or IOBES tagging scheme, we observe that a softmax classifier produces a high number of invalid tags, e.g. an I- tag starts without a previous B-

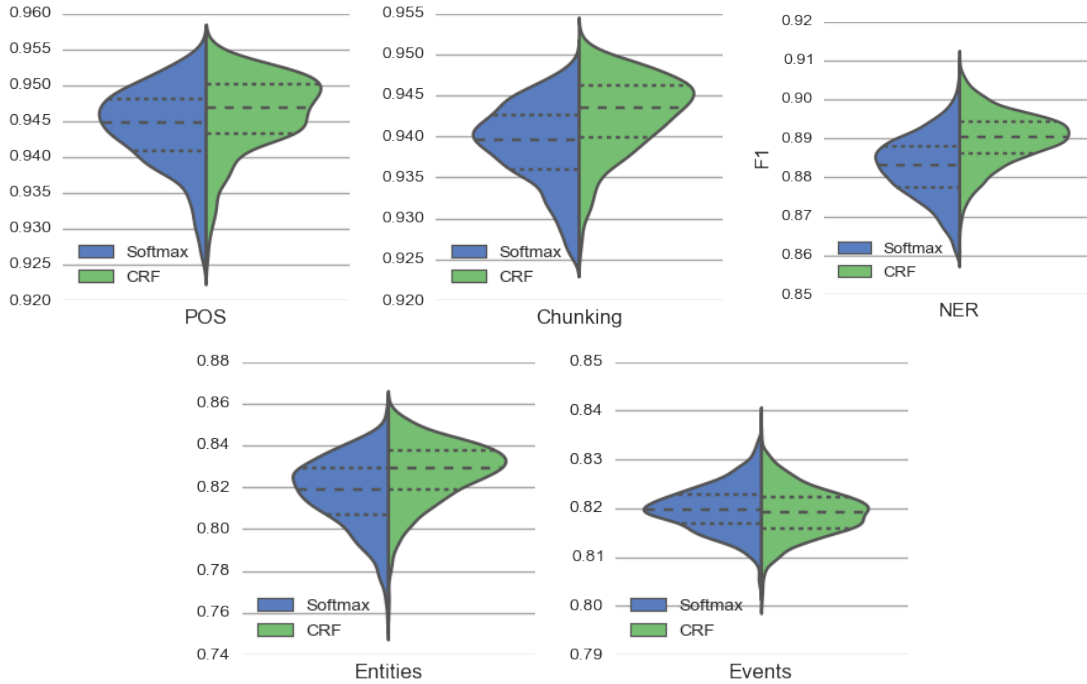


Figure 3.9: softmax versus CRF classifier for the examined benchmark tasks. The plot shows the accuracy for the POS tagging task and the F_1 -score for the other tasks. The violin plots show better results for the CRF classifier for all except the TempEval3 events task.

tag. For example, a system with a single BiLSTM-layer and a softmax classifier produced on the test set for around 1.5% - 2.0% of the named entities an invalid BIO tagging. Increasing the number of BiLSTM reduces the number of invalid tags: A system with three BiLSTM-layers produced invalid tags for around 0.3% - 0.8% of the named entities. We evaluated two strategies for correcting invalid tags: Either setting them to 0 or setting them to the B- tag to start a new segment. However, the difference between these two strategies is negligible. The CRF classifier, on the other hand, produces in most cases no invalid tags and only in rare case are one or two named entities wrongly tagged.

Figure 3.10 depicts the difference between a softmax and a CRF classifier for different numbers of stacked BiLSTM-layers for the NER task. The violin plot shows that a CRF classifier brings the largest improvement for shallow BiLSTM-networks. With an increasing number of BiLSTM-layers, the difference decreases. However, for depth 3 we still observe in the plot a significant difference between the two classifiers. The figure also shows that when using a softmax classifier, more BiLSTM-layers are beneficial, however, when using a CRF classifier, the difference between 1, 2, or 3 BiLSTM layers is much smaller. This effect is further studied in section 3.5.8.

For the TempEval-3 event dataset, softmax is slightly better than a CRF classifier. This is due to the distribution of the labels: The I- tag appears only three times in the whole corpus and not once in the training data. Each event in the training data is therefore composed of only a single token. Hence, except for the tree events in the test dataset, there are no dependencies between the tags and a CRF classifier

Task	# Configs	Softmax	CRF
POS	114	19.3%	80.7% †
$\Delta Acc.$		-0.19%	
σ		0.0149	0.0132
Chunking	230	4.8%	95.2% †
ΔF_1		-0.38%	
σ		0.0058	0.0051
NER	235	9.4%	90.6% †
ΔF_1		-0.67%	
σ		0.0081	0.0060†
Entities	214	13.1%	86.9% †
ΔF_1		-0.85%	
σ		0.0157	0.0140
Events	203	61.6% †	38.4%
ΔF_1			-0.15%
σ		0.0052	0.0057
Average		21.6%	78.4%

Table 3.11: Network configurations were sampled randomly, and each was evaluated with each classifier as a last layer. The first number in a cell depicts in how many cases each classifier produced better results than the others. The second number shows the median difference to the best option for each task. Statistically significant differences with $p < 0.01$ are marked with †.

does not bring any improvement.

Conclusion. In the case there are dependencies between the labels, a CRF classifier usually outperforms a softmax classifier as the last layer of the neural network. This is also the case for stacked BiLSTM-layers. In the case there are no or only negligible dependencies between labels in a sentence, a softmax classifier performs better than the CRF classifier.

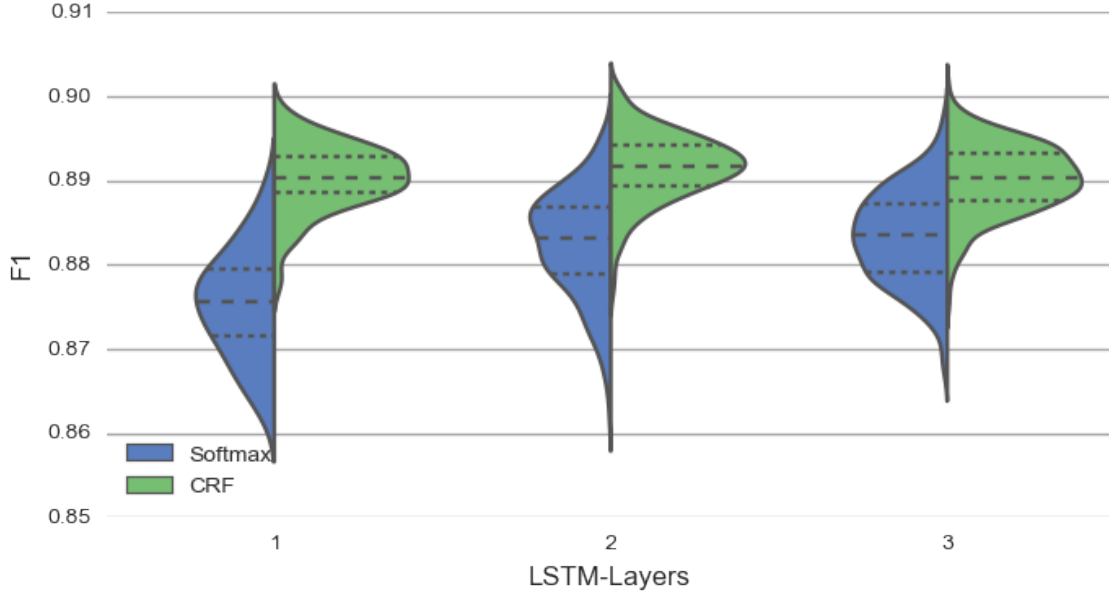


Figure 3.10: Difference between softmax and CRF classifier for different numbers of BiLSTM-layers for the CoNLL 2003 NER dataset.

3.5.7 Dropout

We compare *no dropout*, *naive dropout*, and *variational dropout* (Gal and Ghahramani, 2016). Naive dropout applies a new dropout mask at every time step of the LSTM-layer. Variational dropout applies the same dropout mask for all time steps in the same sentence. Further, it applies dropout to the recurrent units. We evaluate the dropout rates $\{0.05, 0.1, 0.25, 0.5\}$.

Table 3.12 depicts the results for the different dropout schemes. We observe, that variational dropout results in most cases to the best performance. The median difference to not using dropout can be as high as $\Delta F_1 = -1.98\%$ for the Entities task and $\Delta F_1 = -1.32\%$ in comparison to naive dropout. For all tasks it yielded the lowest standard deviation, indicating that variational dropout makes the network more robust in terms of the selected hyperparameters and/or the random seed value.

Table 3.13 evaluates variational dropout, which is applied either to the output or the recurrent units. We conclude that variational dropout should be applied to both units.

Conclusion. Variational dropout was on all tasks superior to no-dropout or naive dropout. Applying dropout along the vertical as well as the recurrent dimension achieved on all benchmark tasks the best result. A theoretical justification for this is provided by Gal and Ghahramani (2016) and our experiments can confirm it.

Task	# Configs	No	Naive	Variational
POS $\Delta Acc.$ σ	77	6.5% -0.27% 0.0083	19.5% -0.18% 0.0108	74.0%† 0.0076
Chunking ΔF_1 σ	91	0.0% -0.88% 0.0055	4.4% -0.53% 0.0053	95.6%† 0.0037†
NER ΔF_1 σ	127	3.9% -0.79% 0.0077	7.9% -0.54% 0.0075	88.2%† 0.0059
Entities ΔF_1 σ	90	2.2% -1.98% 0.0159	6.7% -1.32% 0.0155†	91.1%† 0.0119†
Events ΔF_1 σ	97	15.5% -0.47% 0.0054	17.5% -0.28% 0.0051	67.0%† 0.0038
Average		5.6%	11.2%	83.2%

Table 3.12: Network configurations were sampled randomly, and each was evaluated with each dropout scheme. The first number in a cell depicts in how many cases each dropout scheme produced better results than the others. The second number shows the median difference to the best option for each task. Statistically significant differences with $p < 0.01$ are marked with †.

Task	# Configs	Output	Recurrent	Both
POS $\Delta Acc.$ σ	95	3.2% -0.29% 0.0139	37.9%† -0.05% 0.0119	58.9%† 0.0171
Chunking ΔF_1 σ	163	14.1% -0.32% 0.0050	18.4% -0.25% 0.0053	67.5%† 0.0050
NER ΔF_1 σ	144	9.7% -0.42% 0.0074	22.9% -0.34% 0.0075	67.4%† 0.0063
Entities ΔF_1 σ	144	9.7% -0.82% 0.0149	25.0% -0.64% 0.0142	65.3%† 0.0113
Events ΔF_1 σ	158	29.7% -0.15% 0.0048	15.8% -0.33% 0.0042†	54.4%† 0.0034†
Average		13.3%	24.0%	62.7%

Table 3.13: Network configurations were sampled randomly, and each was evaluated with different dropout rates for variational dropout. The first number in a cell depicts in how many cases each variational dropout scheme produced better results than the others. The second number shows the median difference to the best option for each task. Statistically significant differences with $p < 0.01$ are marked with †.

3.5.8 Going deeper - Number of LSTM-Layers

We sampled hyperparameters and selected randomly a value $60 \leq u \leq 300$ that is divisible by 2 and 3. We then trained one network with a single BiLSTM layer, one network with two BiLSTM layers, and one with three BiLSTM layers. The recurrent units per LSTM were set to $u/\#layers$, for example, with $u = 150$ and two layers, we have two BiLSTM-layers, each of the four LSTMs has 75 recurrent units.

The result is depicted in Table 3.14. For POS-tagging, one and two layers performed the best, for Chunking and NER, two or three layers performed the best, for the Entities tasks two layers performed best and for the Events task, there is no large enough difference between these three options. In conclusion, two BiLSTM layers appears a robust rule of thumb for sequence tagging.

Task	# Configs	Num. LSTM-Layers		
		1	2	3
POS	64	51.6% †	46.9%†	1.6%
$\Delta Acc.$			-0.02%	-0.73%
σ		0.0038	0.0034	0.0154
Chunking	92	10.9%	52.2% †	37.0%†
ΔF_1		-0.29%		-0.11%
σ		0.0059	0.0045	0.0042
NER	84	7.1%	54.8% †	38.1%†
ΔF_1		-0.53%		-0.20%
σ		0.0105	0.0082	0.0079
Entities	75	21.3%	52.0% †	26.7%
ΔF_1		-0.72%		-0.34%
σ		0.0152	0.0128	0.0135
Events	73	30.1%	47.9%	21.9%
ΔF_1		-0.11%		-0.20%
σ		0.0050	0.0041	0.0044
Average		24.2%	50.8%	25.0%

Table 3.14: Network configurations were sampled randomly, and each was evaluated with each possible number of stacked BiLSTM-layers. The number of recurrent units is the same for all evaluated depths. The first number in a cell depicts in how many cases each depth produced better results than the others. The second number shows the median difference to the best option for each task. Statistically significant differences with $p < 0.01$ are marked with †.

Conclusion. Two BiLSTM-layers produced the best results. Adding more layers could not improve the performance. Reducing the number to 1 decreased the performance for all except the POS task. When using one LSTM-layer and a softmax classifier, we observed a high number of invalid BIO-tags, indicating that one LSTM-layer is not sufficient to capture dependencies between tags. Adding another LSTM-layer or adding a CRF classifier reduced the number of invalid BIO-tags usually to zero.

3.5.9 Going wider - Number of Recurrent Units

Finding the optimal number of recurrent units is due to the large number of possibilities not straightforward. If the number of units is too small, the network will not be able to store all necessary information to solve the task optimally. If the number is too big, the network will overfit on the training data and we will observe declining test performance. But as the performance depends on many other factors, and as shown in [chapter 5](#), it also depends heavily on the seed value of the random number generator. Simply testing different recurrent unit sizes and choosing the one with the highest performance will result in wrong conclusions.

To still answer the question of the optimal number of recurrent units as well as how large the impact of this hyperparameter is, we decided to use a method that, due to the large number of runs, is robust to noise from other sources. We decided to compute a polynomial regression between the number of recurrent units and the test performance. We chose a polynomial of degree 2, as we expect that the network will have peak performance at some number of recurrent units and performance will decrease if we choose a smaller value (underfitting) or a larger value (overfitting).

For the polynomial regression, we search for the parameters a , b , and c that minimize the squared error:

$$E = \sum_{j=0}^k |p(x_j) - y_j|^2$$

with $p(x) = ax^2 + bx + c$, x_j the average number of recurrent units per LSTM-network, and y_j the performance on the test set for the samples $j = 0, \dots, k$.

[Figure 3.11](#) illustrates the polynomial regression for the NER task with a two stacked BiLSTM-network. Note, the depicted number of recurrent units is the number of units per LSTM-network. As bidirectional LSTM-networks are used, there are in total 4 LSTM-networks, hence, the total number of recurrent units in the network is 4 times higher than depicted on the x-axis.

We use the polynomial $p(x)$ to find analytically the maximum x_{opt} , i.e., the number of recurrent units that give on average the best test performance. We also use the polynomial to determine how large the impact of this hyperparameter is. A flat polynomial (small a value) is rather robust against changes in this parameters. Selecting a non-optimal number is less important, and it would not be worthwhile to optimize this hyperparameter heavily. A steep polynomial (large a value) is more sensitive. A slightly too small or too large number of recurrent units changes the performance significantly. To make this intuitive understandable, we computed $\gamma_{25} = p(x_{\text{opt}} \pm 25) - p(x_{\text{opt}})$, which depicts how much the test performance will decrease if we choose the number of recurrent units either 25 units too small or too large. [Table 3.15](#) summarizes our results.

As the table shows, the number of optimal recurrent units (per direction) depends on the task and the number of stacked BiLSTM units. The optimal values lay at around 100. However, as the value γ_{25} reveals, this hyperparameter has a rather

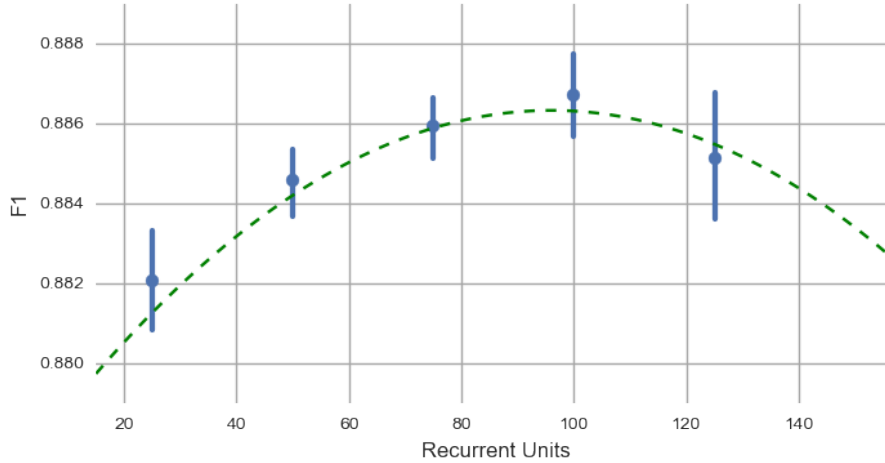


Figure 3.11: Polynomial regression (green, dashed line) for the NER dataset with two stacked BiLSTM-layers. The blue bars are the median and the 95% confidence interval of the median for evenly-spaced bins at different center positions.

small impact on the results. Adding or removing 25 recurrent units from a two stacked BiLSTM-network changes the performance by only roughly 0.01% up to 0.06%.

Task	LSTM-Layers		
	1	2	3
POS	157	63	103
γ_{25}	-0.03%	-0.01%	-0.15%
Chunking	174	106	115
γ_{25}	-0.01%	-0.05%	-0.03%
NER	115	96	92
γ_{25}	-0.01%	-0.06%	-0.07%
Entities	192	175	115
γ_{25}	-0.04%	-0.04%	-0.10%
Events	126	56	-
γ_{25}	-0.01%	-0.03%	-

Table 3.15: The first number in each cell is the optimal number of recurrent units x_{opt} per LSTM-network. The second number shows the value $\gamma_{25} = p(x_{\text{opt}} \pm 25) - p(x_{\text{opt}})$, i.e., when changing the number of recurrent units by 25, how much does the test performance change. For the Events dataset with 3 stacked BiLSTM-layers, the optimal number was not in the tested range and hence was not found by the polynomial regression approach.

Conclusion. The number of recurrent units, as long as it is not far too large or far too small, has only a minor effect on the results. A value of about 100 for each LSTM-network appears to be a good rule of thumb for the tested tasks. This shows, that the capacity of the neural network is of minor importance for the performance. Networks with more capacity (more recurrent units) work as well as networks with less capacity. Other factors, like the pre-trained embeddings, the optimizer, or the

dropout method, are of higher importance for the performance of the network.

3.5.10 Mini-Batch Size

We evaluated the mini-batch sizes 1, 8, 16, 32, and 64. The results are depicted in Table 3.16. It appears that for tasks with small training sets a smaller mini-batch size of 1 up to 16 is a good choice. For larger training sets appears 8 - 32 a good choice. The largest difference was seen for the ACE 2005 Entities dataset, where changing the mini-batch size to 1 decreased the median performance by 2.83 percentage points compared to a mini-batch size of 32.

Task	# Configs	Mini-Batch Size				
		1	8	16	32	64
POS	102	51.0% †	27.5%†	9.8%	5.9%	5.9%
$\Delta Acc.$			-0.07%	-0.16%	-0.26%	-0.20%
σ		0.0169	0.0164	0.0175	0.0183	0.0179
Chunking	94	9.6%	40.4% †	27.7%†	16.0%	6.4%
ΔF_1		-0.56%		-0.05%	-0.10%	-0.22%
σ		0.0556	0.0089†	0.0092†	0.0092†	0.0091†
NER	106	5.7%	16.0%†	22.6%†	30.2% †	25.5%†
ΔF_1		-1.11%	-0.27%	-0.18%		-0.10%
σ		0.0686	0.0120†	0.0096†	0.0090†	0.0099†
Entities	107	2.8%	23.4%†	25.2%†	33.6% †	15.0%
ΔF_1		-2.83%	-0.21%	-0.07%		-0.31%
σ		0.0793	0.0168†	0.0157†	0.0159†	0.0170†
Events	91	33.0% †	25.3%†	27.5%†	6.6%†	7.7%
ΔF_1			0.00%	0.09%	-0.32%	-0.46%
σ		0.0253	0.0144	0.0133	0.0130	0.0139
Average		20.4%	26.5%	22.6%	18.5%	12.1%

Table 3.16: Network configurations were sampled randomly, and each was evaluated with different mini-batch sizes. The first number in a cell depicts in how many cases each mini-batch size produced better results than the others. The second number shows the median difference to the best option for each task. Statistically significant differences with $p < 0.01$ are marked with †.

Conclusion. The size of the mini-batch can decide to which type of minima the network converges. Not all minima generalizes equally well to unseen data (Keskar et al., 2016). Smaller batch sizes typically favor flat minima, which generalize better to unseen data. Larger mini-batches tend to converge to sharper minima, achieving poorer results on unseen data (Hochreiter and Schmidhuber, 1997b; Keskar et al., 2016). However, our results show, that a mini-batch size of 1 can be unfavorable. For the tasks of Chunking, NER, and Entities recognition, a size of 1 significantly reduced the achieved performance. A larger mini-batch size was required.

3.6 Discussion of Evaluation Results

Bergstra and Bengio (2012) analyzed seven design choices and hyperparameters for a feedforward network on various image datasets. They showed that a small number of parameters make the difference between mediocre and state-of-the-art performance. However, the relative importance of each hyperparameter varies between the datasets. Certain parameters were highly relevant for one dataset, but not for other datasets. This varying importance is a challenge for tuning the network for new tasks, as it is a priori not known which parameters will be important, hence, all parameters must be tuned.

In our experiments, we could confirm that only some parameters are relevant for achieving a good performance. For example, we observed large performance differences between the different pre-trained word embeddings, but only minor differences for the number of recurrent units. Further, we observed that several techniques can significantly boost the performance, but their hyperparameters are of minor importance. For example, gradient normalization improved the performance for all tasks, but the normalization threshold τ was of minor relevance. Similar with variational dropout, using it improved the performance, but the dropout rate p was not that important.

In contrast to Bergstra and Bengio (2012), the relative importance of each parameter was comparably consistent across all tasks. Different pre-trained embeddings had a large impact for all tasks, and the number of recurrent units had a minor impact for all tasks. Only for two parameters, we observed changing relative importance: The classifier (Softmax or CRF) and character-based word representations.

The CRF classifier did not yield an improvement for tasks with no or low dependencies between tags, which was expected. For tasks that had stronger dependencies between tags, the CRF classifier yielded a significant improvement. We observed that there was still a difference between a CRF and a softmax classifier with stacked LSTM-layers, even though the difference decreases with more LSTM-layers. It appears that stacked layers are better in capturing dependencies between words in a sentence.

The two character-based word representation mechanisms proposed by Ma and Hovy (2016) and Lample et al. (2016) yielded a statistically significant improvement only for the POS, chunking, and event detection task. For NER and entity recognition, no statistically significant difference was observed. This is contrasting the conclusions from Ma and Hovy (2016) and Lample et al. (2016), which claimed that character-based word representations are helpful for English NER.

Character-based word representations can address two challenges: Creating a meaningful representation for unknown words and usage of sub-word information for the classification, e.g. from morphology. Unknown words, with no pre-trained word embedding, was a minor issue for the analyzed tasks. The ratio of unknown words was only between 0.5% and 3%. For other domains or other languages, unknown words are a bigger challenge, where character-based embeddings could add more value. Sub-word information can be beneficial for syntactical tasks like POS, but for tasks

like NER, they provide lower or no benefit. From the characters of a word, it is often not possible to decide whether it is named entity or which type of entity it is, especially for rare words. The characters in (unseen) stage names, company names or product names usually do not provide much information about the type of entity, hence, deriving a meaningful representation is not possible.

In the experiments of Bergstra and Bengio, the optimal configurations for the different image datasets were fairly distinct, i.e., a value that worked well for one task can be a bad choice for a different task. In our experiments, we observed that the optimal values are rather consistent across the datasets. For example, the embeddings by Komninos and Manandhar (2016) were the best option for all tasks, Nadam (Dozat, 2015) was the optimal optimizer and two stacked BiLSTM-layers achieved the best performance for all datasets or was on-par with the best option.

It is up to future research if this consistency remains true in an evaluation with more diverse datasets, e.g. datasets in different languages or from different domains. A high consistency would be desirable, as it significantly reduces the needed effort of tuning.

Explaining why certain design choices and hyperparameters work well is difficult. The parameter that had the largest impact was the pre-trained word embeddings. This was an important factor for all datasets. High-quality embeddings allow the network to use a lot of background knowledge, that is incorporated into the embeddings. For example, word embeddings can provide information about syntactical and semantic relationships between words. This is especially beneficial for tokens that are not observed during training.

In our experiment, we can confirm that embeddings based on dependencies better capture functional properties of words and window based embeddings capture better topical similarity of words (Levy and Goldberg, 2014; Komninos and Manandhar, 2016). For example, the dependency based embeddings by Levy and Goldberg (2014) worked well for part-of-speech tagging and chunking, but less well for NER or entity recognition. GloVe embeddings (Pennington et al., 2014), which are based on context windows, worked well for NER. Komninos and Manandhar (2016) combined the idea of dependency-based embeddings with a context-based approach. As it appears, this gives a good representation of the functional and of the semantic properties of words. As a consequence, these embeddings performed well in the evaluated tasks.

Our results in section 3.5.9 show, that the capacity of the network, i.e., the number of recurrent units, is of minor importance. The BiLSTM-CRF architecture performs similarly well if we choose the capacity too small or too large. Other aspects of the architecture were far more important: The incorporation of background knowledge using word embeddings and the process to find a local minimum. To which local minimum the network converges is influenced by the optimizer, the dropout mechanism, gradient normalization, and the mini-batch size.

The mini-batch size can influence whether the network converges to a flat or a sharp minimum (Keskar et al., 2016). Hochreiter and Schmidhuber (1997b) (informally) defined that a minimum can be flat, when the error function remains approximately

constant for a large connected region in weight-space, or it can be sharp, when the error function increases rapidly in a small neighborhood of the minimum. A conceptual sketch is given in Figure 3.12. The error functions for training and testing are typically not perfectly synced, i.e., the local minima on the train or development set are not the local minima for the held-out test set. A sharp minimum usually depicts poorer generalization capabilities, as a slight variation results in a rapid increase of the error function. Hence, it is desirable that a network converges to a flat minima, as those usually generalize better to unseen data (Keskar et al., 2016).

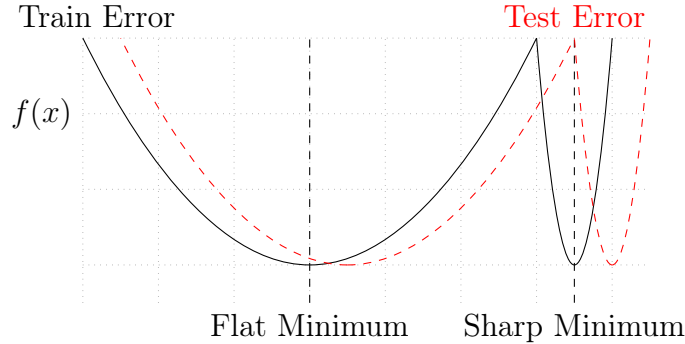


Figure 3.12: A conceptual sketch of flat and sharp minima from Keskar et al. (2016). The Y-axis indicates values of the error function and the X-axis the weight-space.

Keskar et al. (2016) observed that a neural network tends to converge to sharp minima when it is trained with large mini-batches⁹. However, when the network is trained with small mini-batches, it tends to converge to flat minima. They conclude that training with small mini-batches is favorable to achieve better performances. We conclude from our results in section 3.5.10 that training a network with a too small mini-batch size, namely with a size of 1, can also be a disadvantage. For the tasks of Chunking, NER, and Entities recognition, a mini-batch size of 1 achieved far worse results than training the same network with larger mini-batches, for example mini-batches of 8 or 16 sentences. However, for the tasks of POS tagging and event detection, a mini-batch size of 1 was optimal. So far, it is not clear why small mini-batches sizes lead to far worse results on some datasets. How to determine the optimal batch size is part of future research.

3.7 Evaluation on Event Detection Tasks

Using the evaluation results from of section 3.5, we derive a recommended configuration that is depicted in Table 3.17. We evaluate this configuration on five event detection and classification datasets.

We use the following five event datasets: TempEval-3 dataset (UzZaman et al., 2013), ACE 2005 dataset (ACE, 2005) with the split of Li et al. (2013), TAC 2015 dataset (Linguistic Data Consortium, 2015) dataset, ECB+ dataset (Cybulska and

⁹ They used 10% of the available data as batch size.

Parameter	Value
Word Embeddings	Komninos and Manandhar (2016)
Character-based Word Representations	Ma and Hovy (2016)
Optimizer	Nadam
Gradient Normalization	$\tau = 1$
Encoding	BIO
Classifier	CRF
Dropout	Variational with $p = (0.5, 0.5)$
Layers	2
Recurrent Units	100 per layer
Mini-batch Size	8

Table 3.17: Recommended configuration for the BiLSTM-CRF architecture.

Vossen, 2014), and the Richer Event Description (RED) dataset (Palmer et al., 2016). Table 3.18 depicts the number of event mentions for the train, development and test set. A detailed description of these datasets is provided in chapter 2. For the TempEval-3, ECB+ and RED datasets, only the occurrence of an event must be detected without further classification. For the ACE 2005 and TAC 2015 datasets, the task requires the correct classification of the semantic type of the event. The ACE 2005 dataset has 33 event types, and the TAC 2015 has 38 event types.

Dataset	#Events			Task
	Train	Dev	Test	
TempEval-3	9867	532	746	Event detection
RED	6956	851	847	Event detection
ECB+	4386	743	1701	Event detection
TAC 2015	4560	1644	5862	Event detection & classification
ACE 2005	4396	505	416	Event detection & classification

Table 3.18: Overview of corpora used to evaluate the BiLSTM-CRF architecture for the task of event detection.

On each dataset, we trained the architecture with 25 different random seed values and report score distributions in Table 3.19. We observe high F_1 -scores between 79.5% and 86.3% for the TempEval-3, RED and ECB+ dataset. In contrast to the ACE 2005 and TAC 2015 dataset, they do not limit the annotation of events to specific types. Further, the variance of the test scores is smaller, indicating that different minima of the neural network perform comparably on unseen data. The variance of test scores is discussed in greater detail in chapter 5.

The ACE 2005 and TAC 2015 datasets only annotated events of a certain type. Here, the architecture achieves mean scores of 68.0% and 69.4% for the task of event detection and 63.9% and 56.8% for the task of event detection and classification. Having only certain events annotated creates an additional challenge for a

classifier, as it has to decide whether the event is of a certain type. Often, the differentiation can be subtle, for example, the TAC 2015 dataset differentiates between the transportation of people vs. the transportation of objects.

Dataset	Min	Mean	Max	σ
TempEval-3	82.7%	83.5%	84.1%	0.369
RED	84.6%	86.3%	87.4%	0.550
ECB+	77.6%	79.5%	80.5%	0.587
TAC (detection)	67.6%	69.4%	70.5%	0.769
TAC (detection & class.)	51.0%	56.8%	59.0%	1.478
ACE (detection)	66.0%	68.0%	70.1%	1.231
ACE (detection & class.)	61.4%	63.9%	66.2%	1.265

Table 3.19: Minimal, mean, and maximal test F_1 -scores and standard deviation σ of the BiLSTM-CRF architecture for various event datasets.

Table 3.20 compares the mean scores of the BiLSTM-CRF architecture with the reported scores for the state-of-the-art and the inter-annotator agreements. For the TempEval-3 dataset, the reported inter-annotator agreement is 87%, and the best system in the shared task achieved a performance of 81.0% (UzZaman et al., 2013). For the RED dataset, an inter-annotator agreement of 92.8% is reported (O’Gorman et al., 2016) and for the ECB+ dataset, an agreement of 81.1% is reported (Cybulska and Vossen, 2014).

For the TAC 2015 dataset, an inter-annotator agreement of 72% for event detection and 65% for detection and classification is reported (Song et al., 2016). The best system for event detection from the TAC 2015 shared task achieved a performance of 65.3% and the best system for detection and classification achieved a performance of 58.4% (Mitamura et al., 2015a).

The ACE 2005 dataset is one of the oldest datasets in our evaluation and one of the most well-studied datasets. Mitamura et al. (2015b) report an inter-annotator agreement between the first and second annotation pass of 64.8% for event detection and 62.2% for classification. The annotations from both passes were merged, and conflicts were resolved by an expert annotator, which increased the overall consistency of the annotation.

The state-of-the-art system that uses the split provided by Li et al. (2013) achieves an F_1 -score of 71.0% for event detection and 68.7% for event classification (Yang and Mitchell, 2016). Liu et al. (2017b) report slightly higher scores of 70.7% for event detection and classification, but with a different test dataset and using FrameNet as additional training data.

The BiLSTM-CRF architecture performs comparably well for event detection for the TAC and the ACE datasets. On the TAC dataset, it is better than the state-of-the-art, and on the ACE dataset, it is worse by 3 percentage points. However, for the classification of the event type, we observe a larger difference. For the TAC dataset, the difference is 1.6 percentage points, and for the ACE dataset, it is 7.3 percentage points. Recent systems for the ACE dataset, like the systems by Yang and Mitchell

Dataset	BiLSTM-CRF	State-of-the-art	IAA
TempEval-3	83.5%	81.0%	87%
RED	86.3%	-	92.8%
ECB+	79.5%	-	81.1%
TAC (detection)	69.4%	65.3%	72%
TAC (detection & class.)	56.8%	58.4%	65%
ACE (detection)	68.0%	71.0%	64.8%
ACE (detection & class.)	61.4%	68.7%	62.2%

Table 3.20: Performance of the BiLSTM-CRF architecture on the ACE 2005 events dataset in comparison to the state-of-the-art performance and the inter-annotator agreement (IAA).

(2016) and Liu et al. (2017b), perform joint training for the classification of event triggers as well as event arguments. The difference between event types can be subtle and can depend on, e.g., the participants in the event. Hence, joint training and inference on event triggers and event arguments can boost the performance for event classification significantly.

In summary, the presented BiLSTM-CRF architecture performs well for the task of event detection. Without adaption, it achieves state-of-the-art performance or performs comparably well for many datasets for event detection and classification.

3.8 Conclusion

The definition of an *event* depends heavily on the task. This led to many different definitions, annotation guidelines, and datasets. As a consequence, there exists no out-of-the-box system that can be used for all desired use cases. As a consequence, instead of finding an approach that works well for one dataset, we are interested to identify a learning approach that works well for a wide range of event detection datasets.

The BiLSTM-CRF architecture has been proven to work well for many NLP sequence tagging tasks. Hence, it is a strong candidate for a uniform learning approach for event detection. In this chapter, the architecture, as well as different existent design choices for the architecture, were introduced and discussed.

The great flexibility of this architecture creates the challenge of adapting it to new tasks. Tuning of parameters can make the difference between *state-of-the-art* and *mediocre* performance (Hutter et al., 2014). However, only little empirical research has been published so far which aspects of the architecture are critical to tune, and Snoek et al. (2012) described tuning as a “*black art that requires expert experience, unwritten rules of thumb, or sometimes brute-force search*”.

This creates the risk that a lot of time is wasted by tuning the wrong parameters or by implementing unneeded design choices. Further, tuning often requires expert

experiences, making it difficult for non-experts to adapt this architecture to new datasets.

We evaluated ten design options and conclude that most had a rather small impact on the performance. The factors that influenced most the performance were the pre-trained word embeddings, the optimizer, gradient normalization, the dropout mechanism, and the classifier. For the optimizer, Adam (Kingma and Ba, 2014) and Nadam (Dozat, 2015) showed the best performance for all tasks. While performing gradient normalization was important to boost the performance, the concrete threshold value had a minor impact. An optimization of this threshold does not appear worthwhile. Variational dropout (Gal and Ghahramani, 2016) performed best when applied to the output neurons as well as to the recurrent neurons. The CRF classifier consistently showed better performance than a softmax classifier when there were dependencies between the tags. However, the evaluated dataset for event detection does not have strong dependencies between tags, as often only one event trigger is present in a sentence and most triggers consist of a single token.

Other design choices had a rather low impact on the performance. The character-based representations from Lample et al. (2016) and Ma and Hovy (2016) yielded an improvement for only some tasks, and the number of LSTM-layers, as well as the number of recurrent units, had a minor impact on the performance of the network.

Based on the results of our evaluation, we derived a default configuration for the BiLSTM-architecture. It serves as a starting point when applying this architecture to a new dataset.

We applied this configuration to five event datasets and found state-of-the-art or competitive performances for all datasets except for the task of event detection & classification on the ACE 2005 dataset. Here, the difference to the state-of-the-art was 7.3 percentage points. For the ACE 2005 and TAC 2015 datasets, the type of the event can depend on the event arguments. For example, the TAC 2015 dataset distinguishes between the transportation of a person or an object. Recent systems like (Yang and Mitchell, 2016; Liu et al., 2017b) train neural network jointly on event triggers and event arguments. This boosts the performance significantly, but it requires the annotation of event arguments. Adaptation of these approaches to datasets where only the event trigger is annotated is not possible.

We can conclude, that the developed approach works well for the task of event detection and that it can be applied out-of-the-box for various event detection tasks. Classification of events into semantic types can be a challenge, and the incorporation of event arguments can be beneficial to increase the performance.

Chapter 4

Temporal Anchoring of Events

Knowing when an event occurred is necessary for a lot of applications, for example, for time-aware summarization, timeline generation or knowledge base population. In many cases, time plays a crucial role for facts stored in a knowledge base, for example, for the facts when a person was born or died. Also, some facts are only true for a certain time period, like being the president of a country. Event extraction can be used to automatically infer facts for knowledge bases, however, to be useful, it is crucial that the date when the event happened can precisely be extracted.

The TimeBank Corpus (Pustejovsky et al., 2003) is a widely used corpus using the TimeML specifications (Saurí et al., 2004) for the annotations of event mentions and temporal expressions. In order to anchor events in time, the TimeBank Corpus uses the concept of temporal links (TLINKs) that were introduced by Setzer (2001). A TLINK states the temporal relation between two events or an event and a temporal expression. For example, an event could happen *before*, *simultaneous*, or *after* a certain expression of time. The TimeML specifications and the TimeBank Corpus are widely adopted and served as a foundation for several shared tasks, for example for the shared tasks TempEval-1, 2 and 3 (Verhagen et al., 2007, 2010; UzZaman et al., 2013).

The problem with the TimeML specifications is that the number of possible TLINKs scales quadratically with the number of events and temporal expressions. Some documents of the TimeBank Corpus contain more than 200 events and temporal expressions, resulting in more than 20,000 possible TLINKs. Hand-labeling all links is extremely time-consuming and even when using transitive closures and computational support, it is not feasible to annotate all possible TLINKs for a larger set of documents. Therefore, all annotation studies limited the number of TLINKs. For example, in the original TimeBank Corpus, only links that are salient were annotated. Which TLINKs are salient is vague, and the annotation resulted in a comparably low inter-annotator agreement. Furthermore, around 62% of all events do not have any attached TLINK, i.e., for most of the events in the original TimeBank Corpus, no temporal statement can be made.

In contrast to the sparse annotation of TLINKs used in the TimeBank Corpus, the TimeBank-Dense Corpus (Cassidy et al., 2014) used a dense annotation, and all

temporal links for events and time expressions in the same sentence and in directly succeeding sentences were annotated. For a subset of 36 documents with 1,729 events and 289 time expressions, Cassidy et al. annotated 12,715 temporal links, which is around 6.3 links per event and time expression. Besides the large effort needed for a dense annotation, a major downside is the limitation that events and time expressions must be in the same or in adjacent sentences. Our annotation study showed that in 58.72% of the cases the most informative temporal expression for an event is more than one sentence apart from the event mention. For around 25% of the events, the most informative temporal expression is even five or more sentences away. Limiting the TLINKs to pairs that are at most one sentence apart poses the risk that important TLINKs are not annotated and consequently cannot be learned by automated systems.

A further drawback of TLINKs is that it can be difficult or even impossible to encode temporal information that is found in different parts of a text. Given the sentence:

December 30th, 2015 - During New Year's Eve, it is traditionally very busy in the center of Brussels and people gather for the fireworks display. But the upcoming [display]_{Event} was canceled today due to terror alerts.

For a human, it is simple to infer the date for the event *display*. But it is not possible to encode this knowledge using TLINKs, as the date is not explicitly mentioned in the text.

In this chapter, we describe a new annotation scheme to anchor events in time. Instead of using temporal links between events and temporal expressions, we consider the event time as an argument of the event mention. The annotators are asked to write down the date when an event happened in a normalized format for every event mention. An example annotation is depicted in Figure 4.1.

Good evening from the central park in downtown Havana . We are beginPoint=1998-01-19 endPoint=1998-01-23 1998-W04 here this week

1998-01-21 to witness the 1998-01-21 rendezvous that Fidel Castro and the Pope are having with history . We are

beginPoint=1998-01-19 endPoint=1998-01-23 here because what 1998-01-21 happens on this island will also have an

beginPoint=1998-01-21 endPoint=after 1998-01-21 impact on the United States . Everything that happens in Cuba gets America 's

attention . Incidentally , we just show up to do the n/a broadcast , and Cubans n/a want to know what we 're going to n/a tell

Americans , in many cases , what their relatives in the United States are going to n/a hear . Well , this is the eve of the Pope 's

1998-01-21 visit to one of the last bastions of Communism anywhere in the world , and it is already 1998-01-21 causing enormous

Figure 4.1: Sample annotation made with WebAnno. The yellow annotations are existing annotations of temporal expressions from the TimeBank Corpus. The span for the mint annotations, the event mentions, also come from the TimeBank Corpus. Our annotators added the value of the event time for those mint annotations.

Compared to a TLINK annotation, this scheme has several advantages:

- Substantially lower annotation effort. Re-annotating the document from the TimeBank-Dense Corpus required only 1,729 annotations, substantially lower than the 12,715 annotations required for a dense TLINK annotation. Lower annotation effort allows annotating more documents and an easier adaptation of the scheme to new datasets.
- The complete document is taken into account. We show that dense TLINK annotations miss the most important information for the majority of events.
- Temporal information, which is not explicitly stated in the text but is inferred by humans through semantic information, can be annotated.
- The new corpus allows the development and evaluation of automated methods that do not only operate on a sentence level but on a document level.

We show that the annotation is simple for humans, and it can be performed efficiently. However, it poses several challenges for automatic approaches (cf. [section 4.4](#)). To overcome these challenges, we propose a combination of a decision tree combined with neural network classifiers. The system works on the complete document and can extract long-range relations between events and temporal expressions. Further, it can extract begin and endpoints for events that span over multiple days. We show that the proposed model generalizes well to new tasks and textual domains. We applied it without re-training to the SemEval-2015 Task 4 on automatic timeline generation. There, it achieves an improvement of 4.01 points F_1 -score compared to the state-of-the-art. We present the developed system in [section 4.5](#).

The proposed annotation scheme has been published and presented at the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016) ([Reimers et al., 2016b](#)), and the automatic system has been published in the Transactions of the Association for Computational Linguistics (TACL) ([Reimers et al., 2018](#)). The publications are joined work with Nazanin Dehghani during her internship at the UKP Lab.¹

4.1 Previous Annotation Work

Two different annotation styles exist to provide temporal information for events. The first uses the concept of temporal links (TLINKs) that define the relation between two events, two temporal expressions, or an event and a temporal expression. The other method treats the temporal information as an argument for an event and annotates the span within the document.

¹ Nazanin Dehghani was one of the annotators in the annotation study, she ran the CAEVO baseline, helped with computing corpus statistics, and helped with fine-tuning the neural networks in the decision tree. The setup of the annotation study and the annotation guidelines were developed by the author of this thesis, as well as the analysis of the annotation study, the design and implementation of the automatic event time extraction system, the evaluation of the system, and the adaptation of the system to the SemEval-2015 Task 4 dataset. The publications were written by the author of this thesis.

We first present both styles of linking temporal information to events. We discuss the shortcomings of these methods, and why a new method of annotating temporal information for events is needed.

4.1.1 TLINK Based Annotations

A large fraction of corpora, which provide temporal information for events, use temporal links (TLINKs) to anchor events in time. The concept of TLINKs was introduced by [Setzer \(2001\)](#) with the objective to determine the temporal order of events, and, where possible, the calendar date of the event.

A TLINK is defined as the temporal relation between two events, two temporal expressions, or between an event and a temporal expression. As many existent corpora are based on news reports, they often contain a special TLINK that states the relation between the event and the document creation time (DCT). The number of relation classes varies for the different annotation schemes. Setzer proposed five classes: INCLUDED and INCLUDE to mark that an event is temporally included within another event or temporal expression, BEFORE and AFTER to mark that an event happens before or after another event or temporal expression, and SIMULTANEOUS, which she proposed as a fuzzy relation to mark pairs that happen *roughly at the same time*. The TimeML specification² ([Saurí et al., 2004](#)) extended those five classes to 14 classes. Newly added classes were IMMEDIATELY AFTER, IMMEDIATELY BEFORE, IDENTITY, BEGINS, ENDS, BEGUN BY, ENDED BY, DURING and DURING INVERSE.

A well known corpus using this TimeML specification is the TimeBank Corpus ([Pustejovsky et al., 2003](#)). As of writing, the latest version of the TimeBank Corpus was 1.2 with 183 news articles that have been annotated with the TimeML specification.

The 14 temporal relation classes defined by TimeML are quite fine-grained, and it can be challenging for annotators to agree on the right class ([Mani et al., 2006](#)). For example, the difference between BEFORE and IMMEDIATELY BEFORE can be difficult to grasp resulting in disagreement between annotators.

Subsequent corpora often reduced the number of possible relation types. For the shared task TempEval-1 ([Verhagen et al., 2007](#)), the organizers used the event and time annotation verbatim from TimeBank. TLINKs were newly added to this task by seven annotators with a focus on only six relational classes. The same six relational classes were also used for the shared task TempEval-2 ([Verhagen et al., 2010](#)). For the latest shared task on TimeBank, TempEval-3 ([UzZaman et al., 2013](#)), the organizers used 13 relation types, neglecting the DURING INVERSE relation from TimeML.

A challenge for the annotation of TLINKs is the quadratic nature of possible links. The following sentence, that contains two events and two temporal expressions, has six TLINKs:

² Current version: 1.2.1

Mary [left]_{Event} on [Thursday]_{Time} and *John* [arrived]_{Event} [the day after]_{Time}

Given n is the number of events and temporal expressions in a document, the number of possible TLINKs would be $n(n-1)/2$. A mid-sized news article can contain more than 200 events and temporal expressions, which would mean that in theory up to 19,900 TLINKs would be possible. As it is infeasible to annotate all possible relations, different strategies have been used to restrict the number of relations to annotate.

A large fraction of corpora on events use sparse annotations. The TimeBank Corpus (Pustejovsky et al., 2003) has only annotations for salient temporal relations. The subsequent TempEval competitions tried to improve the coverage and added some further temporal links for pairs in the same sentence. However, which relations are *salient* is not clearly defined and was left as a subjective decision to the annotators. This introduces a major dilemma for each unlabeled pair:

1. The annotator missed to look at the pair, hence, a salient relation may or may not exist.
2. The annotator looked at the pair but decided that no salient temporal relation exists.
3. The annotator looked at the pair, but couldn't decide on the correct relation class.

More dense annotations were applied by Bramsen et al. (2006), Kolomiyets et al. (2012), Do et al. (2012) and by Cassidy et al. (2014). Bramsen et al. created directed acyclic graph that encodes temporal relations found in a text by annotating multi-sentence segments of text. Kolomiyets et al. focused on dependency trees of temporal relations where all the events of a narrative are linked via partial ordering relations. Do et al. performed an annotation where “the annotator was not required to annotate all pairs of event mentions, but as many as possible”.

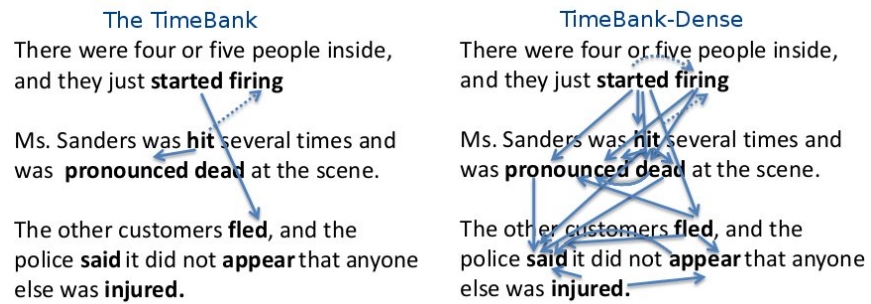


Figure 4.2: Annotation with sparse TLINKs (left) and the same paragraph annotated with a dense annotation (image from Cassidy et al. (2014)).

The densest annotation was performed by Cassidy et al. and was published as TimeBank-Dense Corpus. There, all Event-Event, Event-Time, and Time-Time pairs in the same sentence as well as in directly succeeding sentences were annotated. They adopted from TempEval-1 the **VAGUE** relations for cases where no particular relation can be established. The difference between a sparse annotation, as used

for TimeBank, and a dense annotation, as used for TimeBank-Dense, is illustrated in Figure 4.2. An overview of the different corpora and the number of annotated relations is provided in Table 4.1.

Corpus	Events	Times	TLINKs
TimeBank	7935	1414	6418
TempEval-1	6832	1249	5790
TempEval-2	5688	2117	4907
TempEval-3	11145	2078	11098
Bramsen et al. (2006)	627	-	615
Kolomiyets et al. (2012)	1233	-	1139
Do et al. (2012)	324	232	3132
Cassidy et al. (2014)	1729	289	12715

Table 4.1: Statistics for corpora that use TLINKs.

There are two major drawbacks of dense annotations. First, a large number of relations have to be annotated. For the TimeBank-Dense Corpus, for each event and temporal expression around 6.3 TLINKs had to be annotated. Second, the annotation is limited to only links between expressions in the same or in succeeding sentences. As we will show, for 59% of the events that did not happen at the document creation time, the temporal expression that allows inferring the calendric date of the event is more than one sentence away from the event expression. Hence, a dense TLINK annotation will not include this important relation for most events. As a consequence, a large set of events cannot be anchored temporally, even though for readers it is straightforward to extract this information from the text. Increasing the window size would reduce the number of events that cannot be temporally anchored, however, it results in a significantly increased annotation effort as the number of links grows quadratic.

The specifications of TLINKs from TimeML also have been used in more recent corpora than the TimeBank Corpus. The most recent is the MEANtime corpus (van Erp et al., 2015), that applied a sparse TLINK annotation, and only temporal links between events and temporal expressions in the same and in succeeding sentences were annotated. The MEANtime corpus distinguished between main event mentions and subordinated event mentions and the focus for TLINKs was on main events. The annotation guidelines define 12 different TLINK classes. Further corpora, that are based on TimeML, are the Spanish TimeBank (Sauri and Badia, 2012), the modern Spanish TimeBank (Nieto et al., 2011), and the French TimeBank (Bittar et al., 2011).

4.1.2 Time as Event Argument

The ACE 2005 corpus (Walker et al., 2005), as well as the Rich ERE annotation scheme (Song et al., 2015), defines time as a general event attribute that is tagged if it is within the scope of the corresponding event. The scope of the event is defined as the same sentence that contains the event trigger.

We watched the state funeral in Montreal today for Canada’s former prime minister Pierre Trudeau, who [died]_{Event} [last week]_{Time-Argument} at 80.

For the above sentence, the event *died* would have *last week* as the value for the time argument. While this annotation is simple to perform, it has the shortcoming that the scope is limited to the same sentence.

The temporal information is only given for 19.80% of the events in the ACE 2005 corpus. For all other events, the time argument is empty. The same low number could be observed for the TimeBank-Dense corpus, where only 23.68% of the events had the needed temporal information for temporal anchoring in the same sentence.

Extending the scope of the event might solve the issue that the temporal information for an event is not in the same sentence. However, we observe in our annotation study that for at least 32% of the Single Day Events that there is not a single continuous text passage defining when the event happened. Instead, the temporal anchoring is provided by several text passages scattered throughout the document. This is especially the case for events where the document only provides a rough time frame when the event happened, e.g. the start of the document reveals that it happened after August 1998 and a later text passage reveals that it happened before December 1998.

A corpus that uses a similar idea for the annotation of the event time is the corpus released for SemEval-2015 Task 4 on automatic timeline generation (Minard et al., 2015). The organizers provided annotated news articles on four different topics: Apple Inc., Airbus, General Motors and general stock market news. Events that involved a specific target entity were anchored in time. The format for anchoring was YYYY-MM-DD. By omitting the value DD events could be anchored in a specific month indicating that the event happened at some point within that month. By omitting MM-DD, the event was anchored within a year.

This annotation scheme doesn’t address three key challenges: First, a large set of events last longer than a day, in fact, for the TimeBank-Dense Corpus around 41% of the events lasted longer than a day. The annotation schemes do not provide a notation to specify the begin and end point of such multi-day events. Second, the granularity is either day, month, or year. Specifying that an event happened between November 1st, 2011 and December 31st, 2011 isn’t possible in this scheme. The best anchoring for such an event would be 2011-XX-XX. Worse, if the timeframe overlaps the year boundary, no temporal anchoring is possible. Third, the annotation scheme cannot deal with temporal information stating that something happened before or after a certain date, e.g. that a person was born before 1980. Such temporal information is quite common in news articles, for example for 28% of the events in the TimeBank-Dense corpus, only the information that the event happened before / after a certain date is provided.

4.2 Document-Wide Event Time Annotation Scheme

Our annotation scheme was created with the goal of being able to create a knowledge base from the extracted events in combination with their event times. It assumes that events are already annotated. In our annotation study, we extend the TimeBank-Dense Corpus (Cassidy et al., 2014) with our new annotation scheme.

The TimeBank-Dense Corpus is based on TimeML (Saurí et al., 2004), which defines an *event* as a cover term for situations that *happen* or *occur*. Events can be *punctual* or last for a *period of time*. Predicates describing *states* or *circumstances* in which something holds true are also events. For the TimeBank Corpus, the smallest extent of text (usually a single word) that expresses the occurrence of an event is annotated.

The aspectual type of the annotated events in the TimeBank Corpus can be distinguished into *achievement events*, *accomplishment events*, and *states* (Pustejovsky, 1991). An achievement is an event that results in an instantaneous change of some sort. Examples of *achievement events* are *to find*, *to be born*, or *to die*. *Accomplishment events* also result in a change of some sort, however, the change spans over a longer time period. Examples are *to build something* or *to walk somewhere*. States, on the other hand, do not describe a change of some sort, but that something holds true for some time, for example, *being sick* or *to love someone*. The aspectual type of an event does not only depend on the event itself, but also on the context in which the event is expressed.

Punctual events are a single dot on the time axis while events that last for a period of time have a begin- and an endpoint. It can be difficult to distinguish between punctual events and events with a short duration. Furthermore, the documents typically do not report precise starting and ending times for events. Hence, we decided to distinguish between events that happened at a *Single Day* and *Multi-Day Events* that span over multiple days. We used days as the smallest granularity for the annotation as none of the annotated articles contained any information on the hour, the minute or the second when the event happened. In the case a corpus contains this information, the annotation scheme could be extended to include this information as well.

For *Single Day Events*, the event time is written in the format *YYYY-MM-DD*. For *Multi-Day Events*, the annotator annotates the *begin point* and the *end point* of the event. In the case no statement can be made on when an event happened, the event will be annotated with the label *not applicable*. This applies only to 0.67% of the annotated events in the TimeBank Corpus which is mainly due to annotation errors in the TimeBank Corpus.

*He was **sent** into space on May 26, 1980. He **spent** six days aboard the Salyut 6 spacecraft.*

The first event in this text, **sent**, will be annotated with the event time *1980-05-26*. The second event, **spent**, is a Multi-Day Event and is annotated with the event time *beginPoint=1980-05-26* and *endPoint=1980-06-01*.

In the case the exact event time is not stated in the document, the annotators are asked to narrow down the possible event time as precisely as possible. For this purpose, they can annotate the event time with *after YYYY-MM-DD* and *before YYYY-MM-DD*.

*In 1996 he was **appointed** military attache at the Hungarian embassy in Washington. [...] McBride was **part** of a seven-member crew aboard the Orbiter Challenger in October 1984*

The event **appointed** is annotated *after 1996-01-01 before 1996-12-31* as the event must have happened sometime in 1996. The Multi-Day Event **part** is annotated with *beginPoint=after 1984-10-01 before 1984-10-31* and *endPoint=after 1984-10-01 before 1984-10-31*.

To speed up the annotation process, annotators were allowed to write *YYYY-MM-xx* to express that something happened sometime within the specified month and *YYYY-xx-xx* to express that the event happened sometime during the specified year. Annotators were also allowed to annotate events that happened at the Document Creation Time with the label *DCT*. The full annotation guidelines can be found in the appendix of this thesis. An annotation example is depicted in [Figure 4.1](#).

4.3 Annotation Study

The annotation study was performed on the same subset of documents as used by the TimeBank-Dense Corpus ([Cassidy et al., 2014](#)) with the event mentions that are present in the TempEval-3 dataset ([UzZaman et al., 2013](#)). Cassidy et al. selected 36 random documents from the TimeBank Corpus ([Pustejovsky et al., 2003](#)). These 36 documents include a total of 1498 annotated events. This allows comparing our annotations to those of the TimeBank-Dense Corpus (see section [4.3.6](#)).

Each document has been independently annotated by two annotators according to the annotation scheme introduced above. We used the freely available WebAnno tool ([Yimam et al., 2013](#)). To speed up the annotation process, the existent temporal expressions that are defined in the TimeBank Corpus were highlighted. These temporal expressions are in principle not required to perform our annotations, but the highlighting of them helps to determine the event time. [Figure 4.1](#) depicts a sample annotation. The two annotators were trained on 15 documents distinct from the 36 documents annotated for the study. During the training stage, the annotators discussed the decisions they have made with each other.

After both annotators completed the annotation task, the two annotations were curated by one person to derive one final annotation. The curator examined the events where the annotators disagreed and decided on the final annotation.

4.3.1 Inter-Annotator-Agreement

We use Krippendorff’s α (Krippendorff, 2004) with the nominal metric to compute the Inter-Annotator-Agreement (IAA). The nominal metric considers all distinct labels equally distant from one another, i.e., a partial agreement is not measured. The annotators must therefore completely agree.

Using this metric, the Krippendorff’s α for the 36 annotated documents is $\alpha = 0.617$. Cassidy et al. (2014) reported a Kappa agreement between 0.56 – 0.64 for their annotation of TLINKs. Comparing these numbers is difficult, as the annotation tasks were different. According to Landis and Koch (1977), $\alpha = 0.617$ lies on the border of a moderate and a substantial level of agreement.

4.3.2 Disagreement Analysis

In 648 out of 1498 annotated events, the annotators disagreed on the event time. In 42.3% of the disagreements, the annotators disagreed on whether the event mention is a Single Day Event or a Multi-Day Event. Such disagreement occurs when it is unclear from the text whether the event lasted for one or for several days. For example, an article reported on a meeting, and due to a lack of precise temporal information in the document, one annotator assumed that the meeting lasted for one day, the other that it lasted for several days. A different source for the disagreement has been the annotation of states. They can either be annotated with the date where the text gives evidence that they hold true, or they can be annotated as a Multi-Day Event that begins before that date and ends after that date.

Different annotations for Multi-Day Events account for 231 out of the 648 disagreements (35.6%). In this category, the annotators disagreed on the begin point in 110 cases (47.6%), on the endpoint in 57 cases (24.7%) and on the begin as well as on the endpoint in 64 cases (27.7%). The Krippendorff’s α for all begin point annotations is 0.629, and for all endpoint annotations, it is 0.737.

A disagreement on Single Day Events was observed for 143 event mentions and accounts for 22.1% of the disagreements. The observed agreement for Single Day Events is 80.5% or $\alpha = 0.799$. Most disagreements for Single Day Events were whether the event occurred on the same date as the document was written or if it occurred before the document was written.

4.3.3 Measuring Partial Agreement

One issue of the strict nominal metric is that it does not take partial agreement into account. In several cases, the two annotators agreed in principle on the event time but might have labeled it slightly differently. One annotator might have taken more clues from the text into account to narrow down when an event has happened. One annotator, for example, has annotated an event with the label *after 1998-08-01 before 1998-08-31*. The second annotator has taken an additional textual clue into

account, which was that the event must have happened in the first half of August 1998 and annotated it as *after 1998-08-01 before 1998-08-15*. Even though both annotators agree in principle, when using the nominal metric it would be considered as a distinct annotation.

To measure this effect, we created a relaxed metric to measure mutual exclusivity:

$$d_{ME}(a, b) = \begin{cases} 1 & \text{if } a \text{ and } b \text{ are mutual exclusive} \\ 0 & \text{else} \end{cases}$$

The metric measures whether two annotations can be satisfied at the same time. Given the event happened on August 5th, 1998, then the two annotations *after 1998-08-01 before 1998-08-31* and *after 1998-08-01 before 1998-08-15* would both be satisfied. In contrast, the two annotations *after 1998-02-01* and *before 1997-12-31* can never be satisfied at the same time and are therefore mutually exclusive.

Out of the 648 disagreements, 71 annotations were mutually exclusive. Computing the Krippendorff’s α with the above metric yields a value of $\alpha_{ME} = 0.912$.

4.3.4 Annotation Statistics

Table 4.2 gives an overview of the assigned labels. Around 58.21% of the events are either instantaneous events or their duration is at most one day. 41.12% of the events are Multi-Day Events that take place over multiple days. While for Single Day Events there is a precise date for 55.73% of the events, the fraction is much lower for Multi-Day Events. In this category, only in 19.81% of the cases, the begin point is precisely mentioned in the article, and only in 15.75% of the cases, the endpoint is precisely mentioned.

The most prominent label for Single Day Events is the Document Creation Time (DCT). 48.28% of Single Day Events happened on the day the article was created, 33.49% of these events happened at least one day before the DCT, and 17.43% of the mentions refer to future events. This distribution shows that the news articles and TV broadcast transcripts from the TimeBank Corpus mainly report on events that happened on the same day.

For Multi-Day Events, the distribution looks different. In 76.46% of the cases, the event started in the past, and in 65.10% of the cases, it is still ongoing.

4.3.5 Most Informative Temporal Expression

Not all temporal expressions in a text are of the same relevance for an event. In fact, in many cases, only a single temporal expression is of importance, which is the expression stating when the event occurred. Our annotations allow us to determine *most informative* temporal expression for an event. We define the most informative temporal expression as the expression that has been used by the annotator to determine the event time. We checked for all annotations whether the event date could

	# Events	%
Single Day Events	872	58.21%
with precise date	486	55.73%
after + before	145	16.63%
after	124	14.22%
before	117	13.42%
past events	292	33.49%
events at DCT	421	48.28%
future events	152	17.43%
Multi-Day Events	616	41.12%
precise begin point	122	19.81%
precise end point	97	15.75%
begins in the past	471	76.46%
begins on the DCT	38	6.17%
begins in the future	105	17.05%
ends in the past	179	29.06%
ends on the DCT	26	4.22%
ends in the future	401	65.10%
Not applicable	10	0.67%

Table 4.2: Statistics on the annotated event times. Single Day Events happen on a single day, Multi-Day Events take place over multiple days. The event time can be precise, or the annotators used *before* and *after* to narrow down the event time, e.g. the event has happened in a certain month and year. DCT = Document Creation Time.

be found as a temporal expression in the document and computed the distance to the closest one with a matching value. The distance is measured as the number of sentences. 421 out of 1498 events happened on the Document Creation Time and were excluded from this computation. The Document Creation Time is provided as additional metadata in the TimeBank Corpus, and it is often not explicitly mentioned in the document text.

Figure 4.3 shows the distance between the most informative temporal expression and the event mention. In 23.68% of the cases, the time expression is in the same sentence, and in 17.59% of the cases, the time expression is either in the next or in the previous sentence. It follows that in 58.72%, of the cases the most informative time expression cannot be found in the same or in the preceding or succeeding sentence. This is important to note, as previous shared tasks like TempEval-1,-2, and -3 (Verhagen et al., 2007, 2010; UzZaman et al., 2013) and previous annotation studies like the TimeBank-Dense Corpus (Cassidy et al., 2014) only considered the relation between event mentions and temporal expressions in the same and in adjacent sentences. However, for the majority of events, the most informative temporal expression is not in the same or in the preceding / succeeding sentence.

For 7.31% of the annotated events, no matching temporal expression was found in the document. Those were mainly events where the event time was inferred by the

annotators from multiple temporal expressions in the document. An example is that the year of the event was mentioned in the beginning of the document and the month of the event was mentioned in a later part of the document.

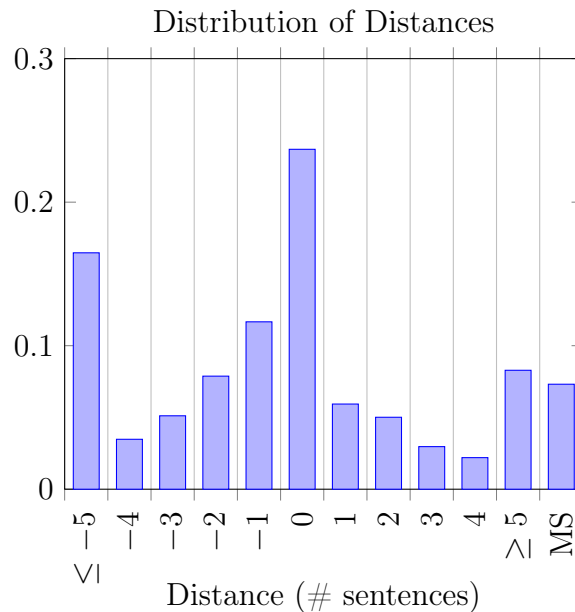


Figure 4.3: Distribution of distances in sentences between the event mention and the most informative temporal expression. For 58.72% of the event mentions, the most informative time expression is not in the same or in the previous/next sentence. For 7.3% of the mentions, the time expression originates from multiple sources (MS).

4.3.6 Comparison of Annotation Schemes

Depending on the application scenario and the text domain, the use of TLINKs or the proposed annotation scheme may be advantageous. TLINKs can capture the temporal order of events, even when temporal expressions are completely absent in a document, which is often the case for novels. The proposed annotation scheme has the advantage that temporal information, independent where and in which form it is mentioned in the document, can be taken into account. However, the proposed scheme requires that the events can be anchored on a time axis, which is easy for news articles and encyclopedic text but hard for novels and narratives.

In this section, we evaluate the application scenario of temporal knowledge base population and time-aware information retrieval. For temporal knowledge base population, it is important to derive the date for facts and events as precisely as possible (Surdeanu, 2013). Those facts can either be instantaneous, e.g. a person died, or they can last for a longer time like a military conflict. Similar requirements are given for time-aware information retrieval, where it can be important to know at which point in time something occurred (Kanhubua and Nørnvåg, 2012).

We use the TimeBank-Dense Corpus (Cassidy et al., 2014) with its TLINKs annotations and compare those to our event time annotations. The TimeBank-Dense

Corpus annotated all TLINKs between Event-Event, Event-Time, and Time-Time pairs in the same sentence and between succeeding sentences as well as all Event-DCT and Time-DCT pairs. Six different link types were defined: **BEFORE**, **AFTER**, **INCLUDES**, **IS_INCLUDED**, **SIMULTANEOUS**, and **VAGUE**, where **VAGUE** encodes that the annotators were not able to make a statement on the temporal relation of the pair.

We studied how well the event time is captured by the dense TLINK annotation. We used transitive closure rules as described by [Chambers et al. \(2014\)](#) to deduct also TLINKs for pairs that were not annotated. For example, when *event*₁ happened before *event*₂ and *event*₂ happened before *date*₁, we can infer that *event*₁ happened before *date*₁. Using this transitivity allows inferring relations for pairs that are more than one sentence apart. For all annotated events, we evaluated all TLINKs, including TLINKs inferred from the transitivity rules, and derived the event time as precisely as possible. We then computed how precise the inferred event times are in comparison to our annotations. Preciseness is measured in the number of days. An event that is annotated with *1998-02-13* has the preciseness of 1 day. If the inferred event time from the TLINKs is *after 1998-02-01 and before 1998-02-15*, then the preciseness is 15 days. A more precise anchoring is preferred.

The TimeBank-Dense Corpus does not have a link type to mark that an event has started or ended at a certain time point. This makes the TLINK annotation impractical for durative events that span over multiple days. According to our annotation study, 41.12% of the events in the TimeBank Corpus last for longer time periods. For these events, it cannot be inferred from when to when the events lasted.

TLINK annotation	# Events	%
same precision	487	55.85%
less precise	198	22.71%
cannot infer time	187	21.44%

Table 4.3: Temporal anchoring based on dense TLINK annotation for Single Day Events in comparison to the temporal anchoring based on our annotation scheme.

Table 4.3 depicts how well the event time can be inferred from a dense TLINK annotation. In 487 out of the 872 Single Day Events (55.85%), TLINKs provide temporal information with the same precision as our annotations. For 198 events (22.71%), they gave a less precise anchoring, i.e., the time window where the event might have happened is larger. For 187 events (21.44%), no event time could be inferred from the TLINKs. This is because there was no link to any temporal expression even when transitivity was taken into account.

For the 487 events where the TLINKs resulted in an event time as precise as our annotation, the vast majority of them were events that happened at the Document Creation Time. As depicted in Table 4.2, 421 events happened at DCT. For those events, the precise date can directly be derived from the annotated link between each event mention and the DCT. For all other events that did not happen at the Document Creation Time, the TLINKs result for the most cases in a less precise

anchoring in time and for around a fifth of these cases in no temporal anchoring at all while we do anchor them.

We can conclude, that even a dense TLINK annotation gives suboptimal information on when events have happened, and due to the restriction that TLINKs are only annotated in the same and in adjacent sentences, a lot of relevant temporal information gets lost.

4.4 Automatic Event Time Extraction

While the new annotation scheme is simple for humans to perform, it poses several challenges for automatic approaches:

1. The number of possible labels is infinite, as date values are part of the labels.
2. Due to the diverse types of events and due to varying temporal information for events, the structure of the labels varies. We have to distinguish between Multi-Day Events and Single Day Events. Further, a text might provide a precise date for the event or only a rough estimate when the event has happened.
3. Temporal information from the whole document must be taken into account to derive the event time. For 58.72% of the events, the most informative time expression is not in the same nor in the neighboring sentences (Figure 4.3), but several sentences away from the event mention.
4. For 7.3% of the events, the event time label is a combination of several temporal clues. In the example of Figure 4.1, the rendezvous event between Fidel Castro and the Pope will happen on the 21st of January. However, this date (1998-01-21) is not mentioned in any temporal expression in the document. Instead, the annotator inferred this date from the document creation time (January 20th) and the phrase “*this is the eve of the Pope’s visit*”.

In order to solve the mentioned challenges, we propose a combination of a decision tree combined with neural network classifiers. The system works on the complete document and can extract long-range relations between events and temporal expressions. Further, it can extract begin and end points for events that span over multiple days.

Existent systems often use complex, handcrafted features. The CAEVO system (Chambers et al., 2014), for example, uses typed dependencies to identify dominant events. Further, it uses WordNet as an external resource to identify synonyms. Systems based on handcrafted features and external resources are often difficult to transfer to new languages or new domains. Our proposed system works end-to-end and does not incorporate any handcrafted features or external resources. Hence, it is simple to train the system on new datasets.

We evaluate the proposed system on our annotated data, where it achieves an accuracy of 42.0% compared to an inter-annotator agreement (IAA) of 56.7%. Compared

to the state-of-the-art CAEVO system (Chambers et al., 2014), we observe a substantial improvement of 33.1 percentage points accuracy for events that happened on a single day. We observe that the system operates better for Single Day Events than for Multi-Day Events. For Single Day Events, the accuracy is at 74.6% (IAA at 80.5%), and for Multi-Day Events, the accuracy is at 24.5% (IAA at 52.0%).

We show that the proposed model generalizes well to new tasks and textual domains. We applied it without re-training to the SemEval-2015 Task 4 on automatic timeline generation. There, it achieves an improvement of 4.01 points F_1 -score compared to the state-of-the-art.

The proposed system has been published in the Transactions of the Association for Computational Linguistics (TACL) (Reimers et al., 2018).

Existent Event Time Extraction Systems

The architecture of approaches usually depends on how temporal information for events is provided in a corpus. The ACE 2005 corpus (Walker et al., 2005) defined time as a general argument for an event and annotated the span that expresses when the event happened (cf. section 4.1.2). Consequently, systems trained and evaluated on the ACE 2005 corpus, extract the event time like other event arguments. A common approach is to formulate this as a pair classification task: Given an event mention and a noun-phrase, a classifier is trained to decide whether the phrase is the time argument for the event. The limitation of such approaches is that the event arguments must be in the same sentence as the event trigger. For only 23.8% of the events in the ACE 2005 corpus is the event time mentioned in the same sentence. For the remaining 76.2% of the events, no temporal information is provided.

Instead of extracting the event time as an argument within the sentence, a large number of systems uses the previously introduced TLINKs (cf. section 4.1.1). A TLINK is defined as the relation between two events, two temporal expressions, or between an event and a temporal expression. The number of possible links grows quadratic with the number of event mentions. As a consequence, all corpora can only provide annotated links for a small subset of possible relations. For example, for the TempEval-3 dataset, 98.2% of the links between events are left unspecified by the annotators (Ning et al., 2017). Typical restrictions for the annotation are either to annotate only salient links and/or to only annotate links within the same sentence or neighboring sentences. As a consequence, automatic approaches, which are trained and evaluated on such corpora, produce labels only for a small subset of links. In a post-processing step, those TLINKs can be used to infer the calendar date for events.

Extracting the relations is often formulated as a pairwise classification task. Each pair of event and/or temporal expression are examined and classified according to the available relational classes. A recent system for dense TLINK extraction was proposed by Chambers et al. (2014). The *CAscading EVent Ordering system* (CAEVO) is a sieve-based-architecture and was trained and evaluated on the TimeBank-Dense Corpus (Cassidy et al., 2014). Chambers et al. describe seven rule-based classifiers

and four machine learning based classifiers. The classifiers are ranked based on their precision.

Temporal graphs are restricted by two rules (Ning et al., 2017):

1. *Symmetry*: If A is *before* B , then B must be *after* A .
2. *Transitivity*: If A is *before* B and B is *before* C , then A must be *before* C .

While symmetry can be ensured by only classifying one direction of the relation, ensuring transitivity can be much more challenging in a pairwise classification task. A system might produce the inconsistent relations that A *before* B , B *before* C , and A *after* C . One frequently used solution is to infer after each classification all temporal relations from transitivity. This strategy is also applied by the CAEVO system, and Chambers et al. report that links from this transitive closure are more precise than the original sieve-provided labels.

Some systems have tried to take advantage of global information to ensure transitivity using Markov logical networks or integer linear programming (Bramsen et al., 2006; Chambers and Jurafsky, 2008; Yoshikawa et al., 2009; UzZaman and Allen, 2010). However, the gains were often minor. A more successful method of taking global information into account was reported by Ning et al. (2017). They developed a structured learning approach where local models are updated based on feedback from global inferences. Compared to previous work, global considerations are not only taken into account in the inference phase but already during the learning phase.

A bottleneck of current systems is the limitation to TLINKs for pairs that are neither in the same nor in adjacent sentences. 28.3% of the events in the TimeBank-Dense Corpus happen at the document creation time (DCT) and can be anchored at DCT using a specialized classifier. For the remaining 71.7% of events, the event time must be inferred via TLINKs. However, for 58.7% of those events the most informative time expression³ is not in the same nor in the previous/next sentence. In conclusion, for 42.1% of all the events in a text, it would be necessary to take long-range TLINKs into account to correctly retrieve the event time. Extending existing systems to take long-range relations into account is difficult due to a lack of training and evaluation data.

4.5 System Architecture

In this section, we present our hierarchical tree approach to automatically infer the event times in a document. In section 4.5.3 we present two baselines that we use for comparison: the first uses dense TLINKs extracted by the CAEVO system and the second baseline is a reduced version of the presented tree approach.

³ The *most informative temporal expression* is defined as the temporal expression that gives the reader the information at which date, or in which time frame, the event happened.

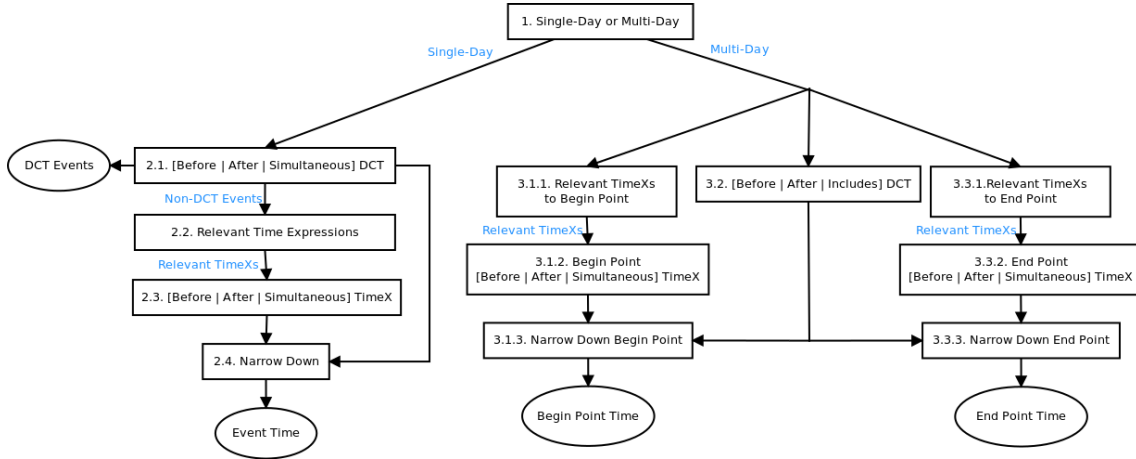


Figure 4.4: Tree structure used to extract the temporal information for an event. Rectangles are local classifiers based on deep convolutional neural networks except for the *Narrow Down* rectangles, which are simple rule-based classifiers.

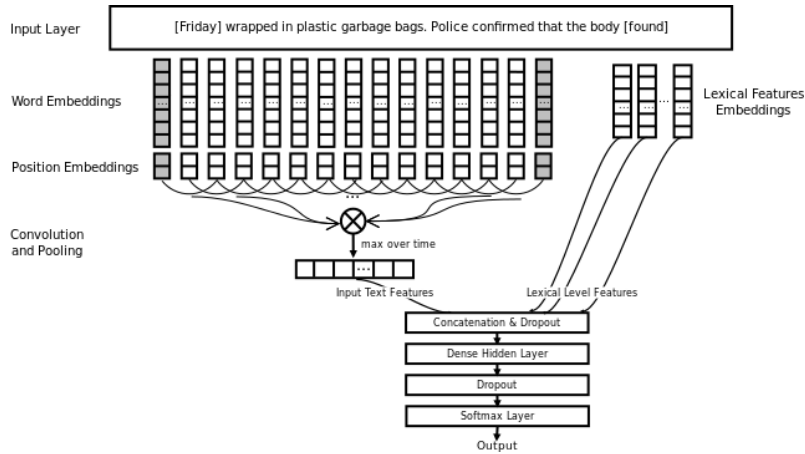


Figure 4.5: The neural network architecture used for the different local classifiers.

4.5.1 Event Time Extraction using Trees

We use the tree structure depicted in Figure 4.4 to extract the event time for a given target event. The structure was inspired by how annotators labeled the data. When annotating the text, the first decision is typically whether the event is a *Single Day Event* or a *Multi-Day Event*. In the case that it is a *Single Day Event*, the next question is whether the event happened at the *Document Creation Time (DCT)* or not. As the annotated data comes from the news domain, a large set of events (48.28% of the *Single Day Events*) happened at the document creation time. In the case the event did not happen at DCT, then the annotator scanned the text to decide whether the date when the event happened is explicitly mentioned or not. If it is not mentioned, the annotator used the *before* and *after* notation to define the time frame when the event happened as precisely as possible. For *Multi-Day Events*, the process is similar to determine the begin and endpoint of the event.

The first classifier is a binary classifier to decide whether the event is a *Single Day* or

a Multi-Day Event. In the case it is a Single Day Event, the next classifier decides the relation between the event and the Document Creation Time (DCT). In the case the event happened at DCT, the architecture stops. If the event happened before or after DCT, the next classifier is invoked, detecting which temporal expressions are relevant for the given event. For all relevant temporal expressions, it is then determined whether the event happened simultaneously, before, or after the temporal expressions. The final step (2.4) outputs a single event time by narrowing down the information it receives from the relation to DCT (2.1) and the pool of relevant temporal expressions and relations (2.3).

For Multi-Day Events the process is similar. However, the system returns the begin point as well as the endpoint. The system runs three processes in parallel: it extracts the relations to relevant time expressions for the begin point (3.1.1 and 3.1.2), it extracts the relation to DCT (3.2), and it extracts the relations to relevant time expressions for the endpoint (3.3.1 and 3.3.2). There are three possible relations between a Multi-Day Event and the DCT: the event started and ended before the DCT; it started and ended after the DCT; or it started before DCT and ended after DCT. This information is taken into account in step 3.1.3 and 3.3.3 when producing single begin point and end point information for the given event.

4.5.2 Local Classifiers

This section describes the different local classifiers applied in our tree structure. For all except the *Narrow Down* classifier, we used the convolutional neural networks Architecture (Lecun, 1989) depicted in Figure 4.5. The *Narrow Down* classifier is a simple, hand-crafted, rule-based classifier described in section 4.5.2.

Neural Network Architecture

We use the same neural network architecture with slightly different configurations for the different local classifiers. The architecture is depicted in Figure 4.5 and is described in the following sections.

The neural network architecture is based on the design proposed by Zeng et al. (2014), which can achieve state-of-the-art performance on relation classification tasks (Zeng et al., 2014; dos Santos et al., 2015). The neural network applies a convolution over the word representations and position embeddings of the input text followed by a max-over-time pooling layer. We call the output of this layer *Input Text Features*. Those Input Text Features are merged with the word embedding for the event and time expression token. The merged input is fed into a hidden layer using either the hyperbolic tangent $\tanh(\cdot)$ or a rectified linear unit (ReLU) as the activation function. The choice of the activation function is a hyperparameter and was optimized on a development set. The final layer is either a single sigmoid neuron, in the case of binary classification, or a softmax layer. To avoid overfitting, we used two dropout layers (Srivastava et al., 2014b), the first before the dense hidden layer and

the second after the dense hidden layer. The percentages of the dropouts were set as hyperparameters.

Word Embeddings. For our experiments, we used the pre-trained word embeddings presented by [Levy and Goldberg \(2014\)](#). The embedding layer of our neural networks maps each token from the input text to their respective word embedding. Out-of-vocabulary tokens are replaced with a special UNKNOWN token, for which the word embedding was randomly initialized.

Position Embeddings. [Collobert et al. \(2011\)](#) proposes the use of position embeddings to keep track how close words in the input text are to certain *target words*. For each input text, we specify certain words as targets. For example, we specify the event and the temporal expression as target words and train the network to learn the temporal relation between those. Each word in the input text is then augmented with the relative distance to the target words. Let pos_1, pos_2, \dots be the positions of the target words in the input text. Then, a word at position j is augmented with the features $j - pos_1, j - pos_2, \dots$. These augmented position features are then mapped in the embedding layer to a randomly initialized vector. The dimension of this vector is a hyperparameter of the network.

The word embeddings and the position embeddings are concatenated to form the input for the convolutional layer. In the case of two target words, the input for the convolutional layer would be:

$$emb_{output} = \{[we^{w_1}, pe^{1-pos_1}, pe^{1-pos_2}], [we^{w_2}, pe^{2-pos_1}, pe^{2-pos_2}], \dots, [we^{w_n}, pe^{n-pos_1}, pe^{n-pos_2}]\}$$

with we^{w_j} the word embedding of the j -th word in the input text, pe^{j-pos_k} the embedding for the distance between the j -th word and the target word k .

Convolutional & Max-Over-Time Layer. A challenge for the classifier is the variable length of the input text and that important information can be anywhere in the input text. To tackle this issue, we use a convolutional layer to compute a distributed vector representation of the input text. Let us define a vector x_k as the concatenation of the word and position embeddings for the position k as well as for m positions to the left and to the right:

$$x_k = ([we^{w_{k-m}}, pe^{k-m-pos_1}, pe^{k-m-pos_2}] || \dots || [we^{w_k}, pe^{k-pos_1}, pe^{k-pos_2}] || \dots || [we^{w_{k+m}}, pe^{k+m-pos_1}, pe^{k+m-pos_2}])$$

The convolutional layer multiplies all x_k by a weight matrix W_1 and applies the activation function component-wise. After that, a *max-over-time* is applied, i.e., the max-function is applied component-wise. The j -th entry of the convolutional & max-over-time layer output is defined as:

$$[conv_{output}]_j = \max_{1 \leq k \leq n} [\tanh(W_1 x_k)]_j$$

Lexical Features. Previous approaches heavily rely on lexical features. For example, the CAEVO system (Chambers et al., 2014) uses, for the classification of event-time edges, the token, the lemma, the POS tag, the tense⁴, the grammatical aspect⁵, and the class of event⁶ as well as the parse tree between event and time expression. In our evaluation, we did not observe that these features have a significant impact on the performance. Hence, we decided to use the event and time tokens as the only features besides the dense vector representation of the input text. For multi-token expressions, we only use the first token.

Our architecture focuses on extracting the event time when event annotations and temporal expressions are provided. In order to evaluate the accuracy of this isolated step, we decided to use the provided annotations in the corpus. The baselines we compared against use these gold annotations as well. Future work includes an end-to-end evaluation that tests those systems with an automated event and temporal expression detection.

Output. The distributed vector representation of the input text and the embeddings of event/time token are concatenated and passed through a dense layer. As the activation function, we allowed either the hyperbolic tangent or the rectified linear unit (ReLU). The choice is a parameter of the network. The final layer is either a single sigmoid neuron, in the case of binary classification, or a softmax layer to compute the probabilities of the different tags.

Single vs. Multi-Day Event Classification

For the first local classifier on deciding whether an event is a *Single Day Event* or a *Multi-Day Event*, we use the sentence as input for the distributed input text representation and the event word as target word.

DCT Classification

News articles often report on events that happened on the same day. Hence, the Document Creation Time (DCT) plays an important role. A large fraction (48.28%) of the Single Day Events happened, for example, on the same day as the document creation time. Both subtrees of the hierarchical classifier use a local classifier to extract the relation between the event and the DCT.

A Single Day Event can happen either before the document was created (**Before**-class), on the same day (**Simultaneous**-class), or it will happen at least one day after the document was created (**After**-class). The configuration of this local classifier is as in the previous section.

⁴ Defined tenses in TimeBank: simple, perfect, and progressive

⁵ Defined aspects in TimeBank: past, present, future

⁶ Defined classes in TimeBank: occurrence, perception, reporting, aspectual, state, i_state, i_action

In the case the event happened on the same day as the document was created, the hierarchical classifier stops. Otherwise, it will continue as described in the next section.

Note, to classify the relation to the DCT, in most cases it was not important to know the concrete Document Creation Time. Therefore, we did not pass the DCT as a value to the network.

For Multi-Day Events, we decided to group the events into three categories: first, events that began and ended before the Document Creation Time (**Before**-class); second, events that began before DCT and ended after DCT (**Includes**-class); and third, events that will begin and end after DCT (**After**-class).

Detecting Relevant Time Expressions

In the case the event did not happen at the DCT, it is important to take the surrounding text and potentially the whole document into account to figure out at which date the event happened. For our classifier, we assume that temporal expressions are already detected in the document. To detect temporal expressions, tools like HeidelTime⁷ can be used that achieve an F_1 -score of 0.919 on extracting temporal expressions in the TimeBank Corpus (Strötgen and Gertz, 2015).

As an intermediate step to detect when an event happened, we first decide whether the temporal expression is relevant for the event or not. We define a temporal expression to be relevant if the (normalized) value of the temporal expression is part of the event time annotation. The value of the temporal expression can be the event time, or it can appear in the **before** or **after** notation.

This step does not yet retrieve the event time annotation nor the temporal relation between the event and the temporal expression. However, it removes irrelevant temporal expressions from the next step in the hierarchical classifier.

The classifier is executed for all event and temporal expression pairs. The input text for the distributed text representation is the text between the event and the temporal expression.

Temporal Relation Classification

Given the relevant temporal expression for an event from the previous step, the next local classifier establishes the temporal relation between the event and the temporal expression. For a given relevant, event-temporal expression pair, it outputs **BEFORE** - when the event happened before the temporal expression, **AFTER** - when it happened after, or **SIMULTANEOUS** - when it happened on the mentioned date. This local classifier has the same configuration as the network used to detect relevant temporal expression.

⁷ <https://github.com/HeidelTime/heideltime>

Narrow Down Classifier

The goal of the Narrow Down Classifier, which is used in step 2.4, 3.1.3 and 3.3.3 in Figure 4.4, is to derive the final label given the information on the relevant temporal expressions, their relation to the event, and the relation to the document creation time. For most events in the corpus, this information was unambiguous, e.g., only one temporal expression was classified as relevant for the event. The proposed approach returns multiple relevant temporal expressions only for a small fraction of the events. However, this number was too small to train and to validate a learning algorithm for this stage. Hence, we decided to implement a straightforward, rule-based classifier. This classifier is depicted in Algorithm 1.

Algorithm 1 Narrow Down Classifier

```

1: function NARROWDOWN(times)
2:   fd_before = FreqDistribution()
3:   fd_after = FreqDistribution()
4:   for [relation, time] in times do
5:     if relation is SIMULTANEOUS then
6:       return time
7:     else if relation is BEFORE then
8:       fd_before.new_sample(time)
9:     else if relation is AFTER then
10:      fd_after.new_sample(time)
11:    end if
12:  end for
13:  //fd_before elements have the fields .num=#samples and .time=time value
14:  if fd_before.size > 0 then
15:    // find the largest number of samples of a time
16:    max_samples = fd_before.max(_.num)
17:    //take minimum over all times having max samples
18:    before_time = fd_before.filter(_.num == max_samples).min(_.time)
19:  end if
20:  if fd_after.size > 0 then
21:    // find the largest number of samples of a time
22:    max_samples = fd_after.max(_.num)
23:    //take maximum over all times having max samples
24:    after_time = fd_after.filter(_.num == max_samples).max(_.time)
25:  end if
26:  return after + after_time + before + before_time
27: end function

```

It takes all relations to relevant temporal expressions as well as the relation to the Document Creation Time to derive the final output. In the case a **SIMULTANEOUS** relation exists, the classifier stops and the appropriate temporal expression is used as event time. If no such relation exists, a frequency distribution of the linked dates and relations is created for **BEFORE** as well as for **AFTER** relations. For example, when the system extracts three relevant **BEFORE** relations of different mentions of **date1** throughout the text and two relevant **BEFORE** relations of different mentions of **date2**, then the system would choose **date1** as a slot-filler for the before property. If there are as many relevant **BEFORE** relations for **date1** as for **date2**, the system will choose the lowest date for the before property (line 14-19). For **AFTER** relations, we use the same logic, except that we choose the largest date (line 24).

4.5.3 Baselines

We use two baselines to compare our system. The first baseline is based on the multi-pass architecture CAEVO introduced by Chambers et al. (2014). Chambers et al. describe multiple rules and machine learning based classifiers to extract TLINKs between events and temporal expressions. This architecture extracts temporal relations of the type BEFORE, AFTER, INCLUDES, IS_INCLUDED, and SIMULTANEOUS. The classifiers are combined into a precision-ranked cascade of sieves. The architecture presented by Chambers et al. does not produce temporal information that an event has started or ended at a certain time point and can therefore only be used for Single Day Events.

The CAEVO system generates a set of <relation, time> tuples in which the event is involved. For example, for the following sentence:

*Police **confirmed Friday** that the body found along a highway*

one sieve adds [IS_INCLUDED, *Friday*₁₉₉₈₋₀₂₋₁₃] and a second sieve adds [BEFORE, *DCT*₁₉₉₈₋₀₂₋₁₄] to the set of possible event times for the *confirmed* event.

If this set contains a relation of type SIMULTANEOUS, IS_INCLUDED or INCLUDES, the baseline sets the event time to the value of the time expression. If the set contains a relation of type BEFORE and/or AFTER, the system narrows down the event time as precisely as possible. When all extracted relations are of type VAGUE, the baseline returns that it cannot infer the time for the event.

Algorithm 2 demonstrates how the event time is selected from this set of possible times.

Algorithm 2 Narrow Down Classifier for CAEVO baseline

```

1: function EVENTTIME(times)
2:   if times is empty then
3:     return 'Not Available'           ▷ the event has no non-vague relation
4:   end if
5:   min_before_time = DATE.MAX_VALUE
6:   max_after_time = DATE.MIN_VALUE
7:   for [relation, time] in times do
8:     if relation is SIMULTANEOUS or IS_INCLUDED or INCLUDES then
9:       return time
10:    else if relation is BEFORE and time < min_before_time then
11:      min_before_time = time
12:    else if relation is AFTER and time > max_after_time then
13:      max_after_time = time
14:    end if
15:  end for
16:  event_time = AFTER + max_after_time + BEFORE + min_before_time
17:  return event_time
18: end function

```

The second baseline is a reduced version of the hierarchical tree presented in the previous sections. For this baseline, we first apply the classifier to decide whether it is a Single Day or Multi-Day Event. When it is a Single Day Event, we classify the relation to the document creation time (DCT) (classifier 2.1). When the event did not happen at DCT, we link it to the closest temporal expression in the document. For Multi-Day Events, we only run the classifier 3.2 to extract the relation to DCT. When the event happened before DCT, we set the begin and end point to **BEFORE** DCT; when it happened after DCT, we set both to **AFTER** DCT; and, when the relation was **Includes**, we set the begin point to **BEFORE** DCT and the endpoint to **AFTER** DCT. In summary, this baseline does not run the classifiers 2.2-2.4, 3.1.1-3.1.3, and 3.3.1-3.3.3 from Figure 4.4, which try to find relevant temporal expressions for none-DCT events.

4.6 Experimental Setup

We conduct our experiments on the TimeBank-EventTime Corpus (cf. chapter 4). The corpus is comprised of 36 documents and 1498 annotated events. We use the same split into training, development, and test set as Chambers et al. (2014) resulting in 22 documents for training, 5 documents for hyperparameter optimization, and 9 documents for the final evaluation. Using this split allows a fair comparison to the CAEVO system.

Hyperparameters for the individual local classifiers were chosen using random search (Bergstra and Bengio, 2012) with at least 1000 iterations per local classifier. The 10 best configurations for each local classifier were evaluated with 25 different random seed values, and the configuration with the highest mean score on the development set was used for the final evaluation on the unseen test set.

4.7 Experimental Results

We evaluate our system using two different metrics. The **strict metric** requires an exact match between the predicted label and the gold label. A disadvantage of this metric is that it does not allow partial agreement. The strict agreement between two annotators is fairly low for events where the exact date of the event was not mentioned.

In order to allow partial matches, we will also use a **relaxed metric**, which will judge two different labels only as an error, if those are mutually exclusive. Two labels are mutually exclusive if there is no event date which could satisfy both labels at the same time. If the event happened on August 5th, 1998, the two annotations *before 1998-08-31* and *after 1998-08-01 before 1998-08-31* would both be satisfied. Therefore, these two different labels would be considered as correct. In contrast, the two annotations *after 1998-02-01* and *before 1997-12-31* can never be satisfied at the same time and are therefore mutually exclusive.

The score of the relaxed metric must be seen in combination with the strict metric. A system could trick the relaxed metric by returning a before date that is far in the future which results in a high relaxed score but a negligible strict score. Future research is necessary to judge the quality of different kind of partial matches and to design an appropriate metric.

4.7.1 System Performance

Table 4.4 presents the mean performance and standard deviation of our system in comparison to the observed inter-annotator agreement (IAA). The mean value and standard deviation is based on 25 training runs with different random seed values.

The inter-annotator agreement is based on two full annotations of the corpus. The chance-corrected agreement is $\alpha = 0.617$ using Krippendorff's α (Krippendorff, 2004). We merged the two annotations into a final gold label annotation of the corpus, which was used for training and evaluation.

	System	IAA
Single Day vs. Multi-Day	78.2% \pm 1.33	81.8%
Single Day Acc. (Strict)	74.6% \pm 1.04	80.5%
Single Day Acc. (Relaxed)	92.5% \pm 0.60	98.0%
Multi-Day Acc. (Strict)	24.5% \pm 1.61	52.0%
Begin Point (Strict)	28.5% \pm 0.73	63.8%
End Point (Strict)	66.5% \pm 1.02	74.9%
Multi-Day Acc. (Relaxed)	74.6% \pm 0.55	94.6%
Begin Point (Relaxed)	94.9% \pm 0.38	98.6%
End Point (Relaxed)	80.2% \pm 0.73	96.1%
Overall Acc. (Strict)	42.0% \pm 1.21	56.7%
Overall Acc. (Relaxed)	84.6% \pm 0.71	95.3%

Table 4.4: Accuracy for the different stages of our system in comparison to the observed inter-annotator agreement (IAA). The *strict* metric requires an exact match between the labels. The *relaxed* metric requires that the two annotations are not mutually exclusive.

The accuracy to distinguish between Single Day and Multi-Day Events is 78.2% on the test set, in comparison to an inter-annotator agreement of 81.8%. The overall performance is 42.0%, compared to an IAA of 56.7% using the strict metric. Using the relaxed metric, the performance is 84.6% for the proposed system in comparison to 95.3% IAA.

For Multi-Day Events, we observe an accuracy with the strict metric of 24.5%, compared to an IAA of 52.0%. Breaking it down to the begin- and end-point extraction, we observe a much lower accuracy for the begin point extraction of just 28.5%, compared to 66.5% accuracy for the end point extraction. However, using the relaxed metric, we see an accuracy of 94.9% for the begin point and 80.2% for the endpoint.

We can conclude that the extraction of the begin point works well, however, in a large set of cases (66.5%) the extracted begin point is less precise than the gold annotation.

We observe a fairly high standard deviation of our system. This is due to the rather small dataset. A few misclassified instances lead to large differences in terms of the performance measures. Further, we observe that the standard deviations are consistently lower for the relaxed metric than for the strict metric.

The baseline based on the CAEVO system from [Chambers et al. \(2014\)](#) can only be applied to Single Day Events, as TLINK types that define the start or the end of an event do not exist. We ran this baseline on all events that were correctly identified as Single Day Events. The performance of this baseline is depicted in Table 4.5. For the proposed approach we observe a performance increase from 41.2% to 74.6%. For 18.3% of the events, the retrieved label of the proposed approach was less precise than the gold label. An example of a less precise label would be *before 1998-12-31* while the gold label was *before 1998-08-15*. A clear wrong label was observed for 7.1% of the generated labels.

A big disadvantage of a dense TLINK annotation is the restriction of TLINKs for events and temporal expressions that are in the same, or in adjacent, sentences. For 32.0% of the events, the baseline was not able to infer any event time information. As our system outputs a label for every event, we see a slightly increased number of wrong labels in comparison to the baseline.

Single Day Events	Ours	CAEVO
Exact match	74.6%	41.2%
Less precise	18.3%	21.5%
Wrong label	7.1%	5.4%
Cannot infer time	-	32.0%

Table 4.5: Distribution of the retrieved labels for the proposed system and for the baseline. The baseline uses a dense TLINK structure and the CAEVO system ([Chambers et al., 2014](#)). *Less precise* are labels where the time frame when the event has happened is larger than for the gold label. *Wrong label* are labels which are in clear contradiction to the gold standard. *Cannot infer time* stands for events where no event time could be retrieved.

Table 4.6 compares the proposed system against the reduced tree that only classifies the type of the event (Single Day or Multi-Day) and the relation to the document creation time. We observe a significant drop in accuracy for Single Day Events, indicating that just classifying the relation to the document creation time is insufficient for this task. The relative drop in performance for Multi-Day Events is lower, indicating that finding relevant temporal expressions and extracting the relations is more challenging for Multi-Day Events than it is for Single-Day Events.

System	SD	MD	Overall
Full system	74.6%	24.5%	42.0%
Reduced tree	40.4%	19.6%	24.2%
CAEVO	41.2%	-	18.1%

Table 4.6: Comparison of the accuracy (strict metric) for Single Day Events (SD), Multi-Day Events (MD) and overall. Reduced tree uses only the local classifiers 1, 2.1 and 3.2.

4.7.2 Error Analysis

Error propagation is an important factor in a decision tree. Table 4.7 depicts the accuracy of the different local classifiers. We compare those to a Majority Vote baseline. For all local classifiers, we can see a large performance increase over the baseline. We observe the lowest accuracy for the classifiers of the begin point (3.1.1. and 3.1.2.). This is in line with the previous observation of the low accuracy for begin point labels as well as with the low IAA for begin point annotations.

	System	Majority Vote
1. Event Type	78.2%	54.5%
Single Day Event		
2.1. DCT Rel.	84.2%	55.6%
2.2. Relevant	79.1%	66.0%
2.3. Relation	81.0%	72.7%
Multi-Day Event		
3.1. Begin Point		
3.1.1. Relevant	79.0%	68.9%
3.1.2. Relation	63.1%	42.9%
3.2. DCT Rel.	65.2%	46.8%
3.3. End Point		
3.3.1. Relevant	83.8%	65.1%
3.3.2. Relation	85.1%	79.0%

Table 4.7: Accuracy for the different local classifiers vs. a Majority Vote baseline. Local classifiers are numbered as depicted in Figure 4.4.

The root classifier, which decides whether the event is a Single Day or a Multi-Day Event, is the most critical classifier. This classifier is responsible for 21.7% of the erroneously labeled events. However, with an accuracy of 78.2% it is already fairly close to the IAA of 81.6%, and it is unclear if this classifier could substantially be improved.

As mentioned in the introduction, the annotators were not restricted to the dates that are explicitly mentioned in the document but could also create new dates. For example, in the sentence *It's the [second day]_{date:1998-03-06} of an [offensive]_{beginPoint=1998-03-05...}* it is clear for the annotator that the offensive started on 1998-03-05. However, this date is not explicitly mentioned in the text, only the date 1998-03-06 is mentioned.

We call such dates *out-of-document* dates. Handling those cases is extremely difficult, and our system is currently not capable of creating such out-of-document dates. Table 4.8 depicts the error rate introduced by those dates.

As the table depicts, 12.6% of the event time labels are affected by out-of-document dates. An especially high percentage of such dates is observed for the begin point of Multi-Day Events. In a lot of these cases, the document states either an explicit or a rough estimation of the duration of the event. In the previous example, the text stated that the offensive already lasted for two days. In another example, the document gives the information that the event started *in recent years* or that it lasted *for roughly 2 1/2 years*.

	Out-of-document dates
Single Day Events	3.0%
Multi-Day Events	24.1%
Begin Point	17.0%
End Point	9.9%
Overall	12.6%

Table 4.8: Percentage of labels in the test set affected by out-of-document dates.

4.7.3 Ablation Test

In this section, we study which components are relevant for the performance of the proposed system.

Table 4.9 presents the changes in accuracy in percentage points when individual components of the proposed system are changed. We observe a slight drop of -2.3 percentage points if bidirectional LSTM-networks with 100 recurrent units are used instead of convolutional neural networks. LSTM-networks showed for other NLP tasks state-of-the-art performance, however, for this task they were not able to improve the performance. One reason could be the comparably small training set of 22 documents. A further disadvantage of the BiLSTM-networks was the significantly longer training time, prohibiting running an extensive hyperparameter tuning.

Configuration	Accuracy
Full system	42.0%
BiLSTM instead of CNN	-2.3
Rnd. word embeddings	-7.7
No input text feature	-9.7
No position feature	-3.9
No narrow down	-1.3

Table 4.9: Change in accuracy (strict metric) in percentage points when replacing individual components of the architecture.

An important factor for the performance was the pre-trained word embeddings. Replacing those with randomly initialized embeddings decreased the performance

by -7.7 percentage points. As before, we think this is due to the small training size. A large number of test tokens do not appear in the training set and several tokens only appear infrequently in the training set. Hence, the network is not able to learn meaningful representations of those words.

Our system successfully uses the text between the event and the temporal expression (*Input Text Features*) for classifying the relation between those. Removing this part of the architecture decreases the accuracy by -9.7 percentage points. Further, it appears that not only the token itself but also the position of the token relative to the event/time token is taken into account. Removing this position information from the input text feature reduces the performance by -3.9 percentage points.

Replacing the narrow down classifier with a classifier that randomly selects one of the relevant temporal expressions reduces the performance by only -1.3 percentage points. For most events, there was only one relevant temporal expression extracted.

We further analyzed the parameter settings for the top five performing local classifiers for each stage. The activation function (*tanh* and *ReLU*) appears to have a negligible impact on the performance. We could find top performing classifiers with *tanh* as well as with *ReLU* as the activation function. For the dropout rate, we noticed strong performance drops if the value was selected too large (larger than $\sim 75\%$) and a slight performance drop if it was too low (lower than $\sim 10\%$). However, a clear pattern between dropout rate and the size of the dense layer could not be observed. A further, more comprehensive study would be needed.

4.7.4 Event Timeline Construction

We evaluated our system on the shared task *SemEval-2015 Task 4: TimeLine: Cross-Document Event Ordering* (Minard et al., 2015). The goal is to construct an event timeline for a target entity given a set of 30 documents from Wikinews on certain topics. We use the setting of Track B, where the events are provided. We used HeidelTime to detect and normalize time expressions. We then ran our system out of the box, i.e., without retraining for the new dataset.

For the shared task, an event can occur either on a specific day, in a specific month, or in a specific year. Events that cannot be anchored in time are removed from the evaluation. We implemented simple rules that transform our system output to the format of the shared task: if an event is simultaneous with a specific time expression, we will output this date. If our system returns that it happened before and after a certain date, it will output the year and month if both dates are in the same month. If both dates are in the same year but in different months, it will output the year. Events with predicted timespans of over more than one year are rejected. For Multi-Day Events, we only use the begin point as only this information was annotated for this shared task.

Two teams participated in the shared task (GPLSIUA and HeidelToul). Currently, the best published performance was achieved by Cornegruta and Vlachos (2016)

with an F_1 -score of 28.58. Our system is able to improve the total F_1 -score by 4.01 points as depicted in Table 4.10.

System	Airbus	GM	Stock	Total
Our approach	30.37	28.83	38.01	32.59
Cornegruta	25.65	26.64	32.35	28.58
GPLSIUA_1	22.35	19.28	33.59	25.36
HeidelToul_2	16.50	10.94	25.89	18.34

Table 4.10: Performance of our system on the SemEval-2015 Task 4 Track B for the topics Airbus, General Motors, and the stock market.

A challenge for our system is the different anchoring of events in time: while our system can anchor events at two arbitrary dates, the SemEval-2015 Task 4 anchors events either at a specific day, month or year. When our system returns the event time value *after 2010-10-01* and *before 2010-11-30*, we had to decide how to anchor this event for the generated timeline. For such an event, three final labels would be plausible: 2010-10-xx, 2010-11-xx, and 2010-xx-xx. A similar challenge occurs for events that received a label like *before 2010-11-30*. If we anchor it in 2010-11-xx, we must be certain that the event happened in November. Similarly, if we anchor it in 2010-xx-xx, we must be certain that the event happened in 2010. Such information cannot be inferred directly from the returned label of our system. As only 30 documents on a single topic were provided for training, we could not tune the transformation accordingly. A manual analysis revealed that this transformation caused around 15% of the errors.

Another source of error was that the system couldn't find any relevant temporal expressions for an event. In that case, the label for the shared task had to be inferred from the relation to the document creation time, which is always returned. This induces the challenge that the system does not return a notion how far away an event is from the document creation time. BEFORE DCT could mean that the event happened yesterday or that the event happened several years in the past.

4.8 Conclusion

We presented a new annotation scheme for anchoring events in time and annotated a subset of the TimeBank Corpus (Pustejovsky et al., 2003) using this annotation scheme. The annotation guidelines as well as the annotated corpus are publicly available.⁸ In the performed annotation study, the Krippendorff's α inter-annotator agreement was considerably high at $\alpha = 0.617$. The largest disagreement resulted from events in which it was not explicitly mentioned when the event happened. Using a more relaxed measure for Krippendorff's α , which only assigns a distance to mutually exclusive annotations, the agreement amounts to $\alpha_{ME} = 0.912$. We conclude that annotators can perform the annotation with a high agreement.

⁸ <https://www.ukp.tu-darmstadt.de/data/timeline-generation/temporal-anchoring-of-events/>

We compared our annotation scheme to the annotation of TLINKs. The effort for annotating TLINKs scales quadratically with the number of events and temporal expressions. To make the annotation feasible, a restriction is often used that only temporal links between events and temporal expressions in the same or in succeeding sentences are annotated. Even with this restriction, the annotation effort is quite significant, as on average 6.3 links per mention must be annotated. As Figure 4.3 depicts, in more than 58.72% of the cases the most informative temporal expression is more than one sentence apart from the event mention. As a consequence, inferring from TLINKs, when an event happened, is less precise, because the temporal information that is more than one sentence away can often not be taken into account.

For the 872 Single Day Events, the correct event time could be inferred from the TLINKs in only 487 cases. For 187 Single Day Events, no event time at all could be inferred, as no temporal expression was within the one sentence window of that event.

A drawback of the proposed scheme is the lack of temporal ordering of events beyond the smallest unit of granularity, which in our case was one day. The scheme is suitable to note that several events occurred at the same date, but their order on that date cannot be encoded. In the case the temporal ordering is important for the application, the annotation scheme could be extended, and TLINKs could be annotated for events that fall on the same date. Another option is to increase the granularity, but this requires that the information is provided in the document.

Automatic Event Time Extraction is a challenging task. The set of labels is infinite, the structure of the label varies depending on the type of the event, and the label depends on information that is scattered across the document. Further, the annotations do not reveal from which part of the document the annotator got the temporal information. It might be that a label is composed out of several temporal clues which were mentioned in different sentences of the document. Hence, modeling this as a pairwise relation classification task is not possible.

The presented classifier is based on a decision tree that applies neural networks at its nodes. The decision tree executes those local classifiers successively, and branching is based on the output of those classifiers. The first stage decides whether the event is a Single Day Event or a Multi-Day Event. The second stage extracts the relation to the document creation time, as a large fraction of the events happened at DCT. If an event did not happen at DCT, a first classifier identifies relevant temporal expressions and a second classifier identifies the relation types between the event and the relevant temporal expressions. The final stage is a rule-based system, which constructs the event time.

The local classifiers for identifying relevant temporal expressions and classifying the type of the relations take the whole document into account. We applied the system to the created TimeBank-EventTime Corpus and achieved an accuracy of 42.0% in comparison to an inter-annotator agreement of 56.7% using a strict metric. For 74.6% of the Single Day events, the exact event time could be extracted. This is a 33.1 percentage points improvement in comparison to the state-of-the-art approach

by Chambers et al. (2014).

An ablation test in section 4.7.3 analyzed which parts of the complex classifier are crucial for the performance of the system. Of special importance was the text input between an event and a temporal expression. Removing this part of the system caused the accuracy to drop by 9.7 percentage points. This large drop in accuracy indicates that the local classifiers successfully take the text into account for their classification decision.

We showed that the system is able to identify which temporal expressions are relevant to an event and that long-range dependencies can be successfully be identified and classified. Further, we demonstrated the generalizability by applying it to the SemEval-2015 Task 4 on timeline generation. Without re-training the network, it was able to improve the F_1 -score by 4.01 percentage points compared to the current state-of-the-art system.

A limitation of the proposed system is that it can only produce date values which are explicitly mentioned in the document. 7.3% of the events in the corpus (12.6% for test set) have *out-of-document* event times. An out-of-document event time is a date that was semantically inferred by the annotators and which is not explicitly mentioned in the document itself. Such dates are often the result of merging several temporal and/or semantic clues together. Extracting those out-of-document dates would require a deep semantical understanding of the text.

Chapter 5

Challenges in Evaluating Machine Learning Approaches

In research, we are often interested to identify the best approach for solving a particular task. In NLP, the tool of choice is in many cases machine learning. An approach is trained on a specific training dataset, and the resulting model is applied to new, unseen data. The fundamental question arises how to compare two or more approaches for a specific task in order to conclude which is the best machine learning approach? When can we say that approach A is superior to approach B for a particular task?

Superior can mean a lot of things, for example, requiring less compute time for training, faster inference speed, requiring less memory, smaller storage requirements or ease of use. While those factors can be critical when applying approaches in real-world applications, research mainly focuses on how accurate different approaches can solve the task and evaluating approaches on common benchmark datasets have become a major driving force in research.

At first glance, the answer, which of two learning approaches is more accurate, appears simple: we train both learning approaches on a certain dataset, maybe tune it on a development set, and then compute a performance score on a held-out test set. The approach with the higher test performance score is observed as superior¹ (cf. [Figure 5.1](#)).

As the test set is a finite sample, the test score differs from the (hypothetical) performance on the complete data distribution. A significance test on the test set is used to reduce the risk that chance, induced from the finite test sample, is the explanation for the difference. It is usually accepted that one approach is superior to the other for that task if the difference is statistically significant.

This evaluation methodology is often used in scientific publications and for shared tasks in the NLP field. For example, it is commonly used for the shared tasks at

¹ In this thesis, we only judge approaches based on how accurate those are, given a specific performance measure. For real-world applications, superiority can mean many distinct things that are not related to accuracy.

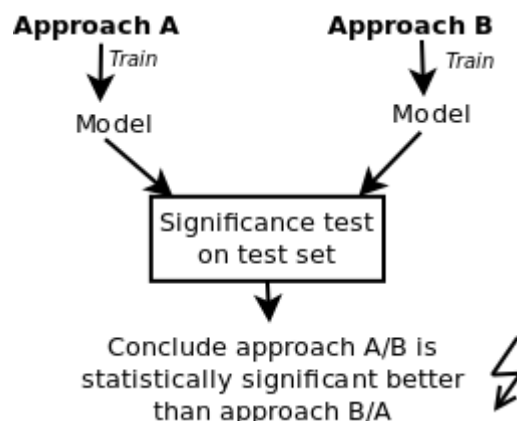


Figure 5.1: Common evaluation methodology to compare two approaches for a specific task.

the International Workshop on Semantic Evaluation (SemEval) and for the shared tasks at the Conference on Computational Natural Language Learning (CoNLL). The participants either submit the output of their system for the unlabeled test data to the task organizers or, as it was the case for the CoNLL 2017 shared task on multilingual parsing (Zeman et al., 2017), participants submitted their system to a cloud-based evaluation platform where it was applied to new data. To identify if differences are significant, the organizers used paired bootstrap resampling. Hiding the test data from the participants eliminates the risk that information about the test data is used for the design of the approach.

The following question arises: How reliable is this evaluation setup to identify better / the best machine learning approach for a certain task?

In this chapter, we show that this commonly used evaluation methodology does not allow to identify superior machine learning approaches. As we will show, a significant difference in test score is not necessarily the result of a superior architecture, but there is a high risk that chance is the explanation for a statistically significant performance difference. The issue is not the significance test, hence, it cannot be solved by selecting a different significance test.

Section 5.2 starts with a formalization of two commonly observed evaluation methodologies used for scientific publications and shared tasks that are based on the comparison of individual test scores. We tested these evaluation methodologies for two event detection tasks and five common sequence tagging tasks with a recent BiLSTM-CRF architecture (chapter 3). The results in section 5.3 show that the evaluation methodologies are unable to identify superior learning approaches. The observation is not limited to the studied tasks as shown in section 5.4.

Parts of the results of this chapter have been published and presented at EMNLP 2017 (Reimers and Gurevych, 2017a) and in (Reimers and Gurevych, 2018).

5.1 Evaluating Learning Approaches vs. Models

In the context of this chapter, it is important to notice the difference between *models* and *learning approach*. A *learning approach* describes the holistic setup to solve a certain optimization problem. For neural networks, this would be the network architecture, the optimization algorithm, the loss-function, etc. A *model* is a specific configuration of the weights for this architecture. Such a model can be applied to new instances.

As described by [Dietterich \(1998\)](#), there are two different sets of questions depending on whether *learning approaches* or *models* are evaluated. In a specific application setting, the goal is often to find the best model for that task. For this model, we like to estimate the accuracy for future samples.

In research, we are often less interested in actually applying a model to new examples. Instead, we are more interested to find the best learning approach for a specific task. Hence, published research papers often focus on the architecture and the applied learning strategy. Usually, they do not focus on the final weights of a specific model.

A commonly used methodology to compare models is described by [Bishop \(2006\)](#) (p. 32): “*If data is plentiful, then one approach is simply to use some of the available data to train a range of models, [...], and then to compare them on independent data, sometimes called a validation set, and select the one having the best predictive performance. [...] it may be necessary to keep aside a third test set on which the performance of the selected model is finally evaluated.*”

We observe that often the conclusion is drawn that a superior model implies a superior learning approach for a task. For example, for the shared task SemEval-2017 on semantic textual similarity (STS), the task organizers conclude that the model from the winning team is “*the best overall system*” ([Cer et al., 2017](#)). [Szegedy et al. \(2015\)](#) conclude that the winning model from Clarifai for the ImageNet 2013 challenge was the “*year’s best approach*”. Further, new design and architectural choices introduced by authors are often justified by showing a higher model performance on common benchmark datasets. For example, [Lample et al. \(2016\)](#) presented an extension to the BiLSTM-CRF model ([Huang et al., 2015](#)) for sequence tagging that derives word representations based on the characters of a word. The benefit of this extension is justified by an increased model performance of +0.74 percentage points F_1 -score for the CoNLL-2003 NER dataset. [Ma and Hovy \(2016\)](#) used convolutional neural networks (CNN) to derive character-level representations and reported increased model performance for the CoNLL 2003 NER dataset and for English POS tagging on the Wall Street Journal portion of Penn Treebank. They draw the conclusion that the “*BLSTM-CNN models significantly outperform the BLSTM model, showing that character-level representations are important for linguistic sequence labeling tasks*”. Further, they draw the conclusion of significant improvements over the architecture proposed by Lample et al. We show that these conclusions, which are based on single test scores, are unjustified.

The purpose of most shared tasks, like the shared tasks organized by the Interna-

tional Workshop on Semantic Evaluation (SemEval) or by the Conference on Computational Natural Language Learning (CoNLL), is to identify the best approaches for a specific task. The participants usually do not have to publish the model, but they have to describe their approach so that it is available for the community. Depending on the prestige of the shared task, winning it can come along with a lot of visibility. The winning approach is often part of future research, e.g., by further tuning or extending it or by applying it to new datasets.

The following sections show that evaluating models and learning approaches are two different things and that a significantly better model performance does not allow to draw the conclusion of a superior learning approach. We show that there is a high risk that chance, and not a superior design, leads to significant differences in model performances. For example, for the CoNLL 2003 shared task on NER, we compared two identical neural networks with each other. In 22% of the cases, we observed a significant difference in test scores with $p < 0.05$. By implication, if we observe a significant difference in test performance, we cannot be certain whether the difference is due to a superior approach or due to luck. The issue is not a flawed significance test but lies in wrongly drawn conclusions. Two or more machine learning approaches cannot be compared based on individual model performances.

5.2 Evaluation Methodologies Based on Single Model Performances

This section formalizes two evaluation methods that are based on performances of single models. These two evaluation methodologies are predominantly used in shared tasks as well as in many scientific publications to show that a new learning approach is superior to existent approaches.

Single Run Comparison

The first evaluation method is to train both approaches a single time and to compare the test scores.

Evaluation 1. Given two approaches, we train both approaches a single time to generate the models A_i and B_j . We define $\Psi_{A_i}^{(Test)}$ as the test score for model A_i and $\Psi_{B_j}^{(Test)}$ as the test score for model B_j . We call approach A is superior to approach B if and only if $\Psi_{A_i}^{(test)} > \Psi_{B_j}^{(test)}$ and the difference is statistically significant. Commonly used significance tests are an approximate randomized test or a bootstrap test (Riezler and Maxwell, 2005).

Non-deterministic learning approaches², like neural networks, can produce many distinct models A_1, \dots, A_n . Which model is produced depends on the sequence of

² We define a learning approach as non-deterministic if it uses a sequence of random numbers to solve the optimization problem. The observations are extendable to deterministic approaches that have tunable hyperparameters.

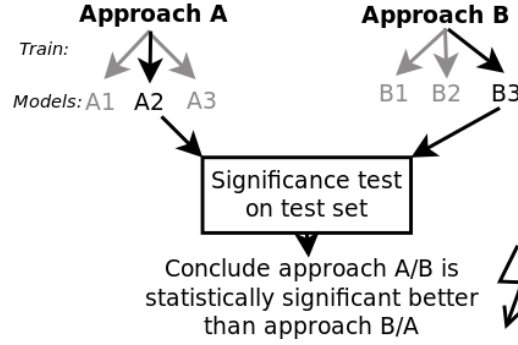


Figure 5.2: Single score comparison for non-deterministic learning approaches (Evaluation 1).

random numbers and cannot be determined a priori.

Figure 5.2 illustrates the issue of this evaluation methodology for non-deterministic learning approaches. Approach *A* produces the model A_2 , while approach *B* the model B_3 . Model A_2 might be significantly better than B_3 , however, it might be worse than the other models B_1 or B_2 .

Training of neural networks is highly non-deterministic as it usually depends on a random weight initialization, a random shuffling of the training data for each epoch, and repeatedly applying random dropout masks. The error function of a neural network is a highly non-convex function of the parameters with the potential for many distinct local minima (LeCun et al., 1998; Erhan et al., 2010). Depending on the seed value for the pseudo-random number generator, the network will converge to a different local minimum. However, not all minima generalize equally well to unseen data (Hochreiter and Schmidhuber, 1997b; Erhan et al., 2010; Keskar et al., 2016). Erhan et al. showed that with increasing depth the probability of finding poor local minima increases.

The issue that different convergence points of a neural network generalize differently to new data has been known before. However, in research, and also when participating in a shared task, we are often interested in the best performance an approach can achieve, e.g., after training the approach multiple times or after tuning the hyperparameters.

Best Run Comparison

For shared tasks, the participants are not restricted to train their approach only once. Instead, they can train multiple models and can tune the parameters on the development set. For the final evaluation, they usually must select one model that is compared to the submissions from other participants. A similar process can often be found in scientific publications, where authors tune the approach on a development set and report the test score from the model that performed best on the development set. This form of evaluation is formalized in the following (depicted in Figure 5.3).

Evaluation 2. Given two approaches and we sample from each multiple models. Approach A produces the models A_1, \dots, A_n and approach B the models B_1, \dots, B_m with sufficiently large numbers of n and m . We define A_* as the best model from approach A and B_* as the best model from approach B . Bishop (2006) defines the best model as the model that performed best on the unseen development set:

$$A_* = \operatorname{argmax}_{A_i \in \{A_1, \dots, A_n\}} (\Psi_{A_i}^{(dev)})$$

$$B_* = \operatorname{argmax}_{B_i \in \{B_1, \dots, B_m\}} (\Psi_{B_i}^{(dev)})$$

With $\Psi^{(dev)}$ the performance score on the development set. We call approach A is superior to approach B iff $\Psi_{A_*}^{(test)} > \Psi_{B_*}^{(test)}$ and the difference is significant.

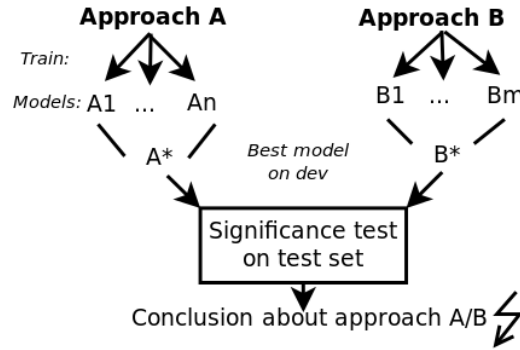


Figure 5.3: Illustration of model tuning and comparing the best models A_* and B_* (Evaluation 2).

In the following sections we show that the conclusion $\Psi_{A_*}^{(test)} > \Psi_{B_*}^{(test)} \Rightarrow \text{approach } A \text{ better than approach } B$ is wrong. This implies that this evaluation methodology is unsuitable for shared tasks and research publications that try to identify superior learning approaches for a task.

5.3 Empirical Study: Comparing Methods Based on Single Model Performances

We demonstrate that Evaluation 1 and Evaluation 2 fail to identify that two learning approaches are the same. By implication, a significant difference in test score does not allow the conclusion that one approach is better than the other.

We compare a learning approach A against itself, which we call approach A and \tilde{A} hereafter. Approach A and \tilde{A} use the same code, with the same configuration and are executed on the same computer. The only difference is that the sequence of random number changes each time the approaches are trained.

A suitable evaluation method should conclude that there is no significant difference between A and \tilde{A} in most cases. We use $p = 0.05$ as the threshold, hence, we would expect that a significant difference between A and \tilde{A} only occurs in at most 5% of the cases.

As datasets we use the datasets described in [section 3.3](#) for common NLP sequence tagging tasks. As learning approach, we use the BiLSTM-CRF architecture described in [section 3.1](#). We use 2 hidden layers, 100 hidden units each, variational dropout ([Gal and Ghahramani, 2016](#)) of 0.25 applied to both dimensions, Nadam as optimizer ([Dozat, 2015](#)), and a mini-batch size of 32 sentences. For the English datasets, we use the pre-trained embeddings by [Komninos and Manandhar \(2016\)](#). For the German datasets, we trained word embeddings using word2vec on about 116 Million sentences from German Wikipedia and German news articles. The German embeddings were published in [Reimers et al. \(2014\)](#).

Training

In total, we trained 100,000 models for each task with different random seed values. We randomly assign 50,000 models to approach A while the other models are assigned to approach \tilde{A} .

For simplification, we write those models as two matrices with 50 columns and 1,000 rows each:

$$[A_i^{(j)}] \quad [\tilde{A}_i^{(j)}]$$

with $i = 1, \dots, 50$ and $j = 1, \dots, 1000$. Each model $A_i^{(j)}$ has a development score $\Psi_{A_i^{(j)}}^{(dev)}$ and test score $\Psi_{A_i^{(j)}}^{(test)}$.

Model $A_*^{(j)}$ marks the model with the highest development score from the row $A_{1 \leq i \leq 50}^{(j)}$ and $\tilde{A}_*^{(j)}$ is the model with the highest development score from $\tilde{A}_{1 \leq i \leq 50}^{(j)}$. Hence, we test Evaluation 2 with $n = m = 50$.

Statistical Significance Test

We use the bootstrap method by [Berg-Kirkpatrick et al. \(2012\)](#) with 10,000 samples to test for statistical significance between test performances with a threshold of $p < 0.05$.

For Evaluation 1, we test on statistical significance between the models $A_i^{(j)}$ and $\tilde{A}_i^{(j)}$ for all i and j . For Evaluation 2, we test on statistical significance between $A_*^{(j)}$ and $\tilde{A}_*^{(j)}$ for $j = 1, \dots, 1000$.

Results

We compute in how many cases the bootstrap method finds a statistically significant difference. Further, we compute the average F_1 test-score difference τ for pairs with an estimated p -value between 0.04 and 0.05. This value can be seen as a threshold: If the F_1 -score difference is larger than this threshold, there is a high chance that the bootstrap method testifies a statistical significance between the two models.

Further, we compute the differences between the test performances for approach A and \tilde{A} . For Evaluation 1, we compute $\Delta^{(test),(i,j)} = |\Psi_{A_i^{(j)}}^{(test)} - \Psi_{\tilde{A}_i^{(j)}}^{(test)}|$. For Evaluation 2, we compute:

$$\Delta^{(test),(j)} = |\Psi_{A_*^{(j)}}^{(test)} - \Psi_{\tilde{A}_*^{(j)}}^{(test)}|.$$

For those delta values we compute a 95% percentile $\Delta_{95}^{(test)}$. The value indicates that a difference in the test score for a given task should be higher than $\Delta_{95}^{(test)}$, otherwise there is a chance greater 5% that the difference is due to chance for the given task and the given network architecture.³

Single Run Comparison

Table 5.1 depicts the main results for Evaluation 1. For the *ACE 2005 - Events* task, we observe in 34.48% of the cases a significant difference between the models $A_i^{(j)}$ and $\tilde{A}_i^{(j)}$. For the other tasks, we observe similar results and between 10.72% and 33.20% of the cases are statistically significant.

The average F_1 -score difference for statistical significance for the *ACE 2005 - Events* task is $\tau = 1.97$ percentage points. However, we observe that the difference between $A_i^{(j)}$ and $\tilde{A}_i^{(j)}$ can be as large as 9.04 percentage points F_1 . While this is a rare outlier, we observe that the 95% percentile $\Delta_{95}^{(test)}$ is more than twice as large as τ for this task and dataset.

Task	Threshold τ	% significant	$\Delta_{95}^{(test)}$	$\Delta_{Max}^{(test)}$
ACE 2005 - Entities	0.65	28.96%	1.21	2.53
ACE 2005 - Events	1.97	34.48%	4.32	9.04
CoNLL 2000 - Chunking	0.20	18.36%	0.30	0.56
CoNLL 2003 - NER-En	0.42	31.02%	0.83	1.69
CoNLL 2003 - NER-De	0.78	33.20%	1.61	3.36
GermEval 2014 - NER-De	0.60	26.80%	1.12	2.38
TempEval-3 - Events	1.19	10.72%	1.48	2.99

Table 5.1: The same BiLSTM-CRF approach was evaluated twice under Evaluation 1. The threshold column depicts the average difference in percentage points F_1 -score for statistical significance with $0.04 < p < 0.05$. The % *significant* column depicts the ratio how often the difference between $A_i^{(j)}$ and $\tilde{A}_i^{(j)}$ is significant. Δ_{95} depicts the 95% percentile of differences between $A_i^{(j)}$ and $\tilde{A}_i^{(j)}$. $\Delta_{Max}^{(test)}$ shows the largest difference.

We observe those variances not only for our implementation of the BiLSTM-CRF architecture. We observe this issue also for two recently published BiLSTM-CRF systems for Named Entity Recognition from Ma and Hovy (2016) and from Lample et al. (2016). Lample et al. reported an F_1 -score of 90.94% and Ma and Hovy reported an F_1 -score of 91.21% for English NER. Ma and Hovy draw the conclusion

³ Note that $\Delta_{95}^{(test)}$ depends on the used machine learning approach and the specific task.

that their system achieves a significant improvement over the system by Lample et al.

We re-ran both implementations multiple times, each time only changing the seed value of the random number generator. We ran the Ma and Hovy system 86 times and the Lample et al. system, due to its high computational requirement, for 41 times. The score distribution is depicted as a violin plot in Figure 5.4. Using a Kolmogorov-Smirnov significance test (Massey, 1951), we observe a statistically significant difference between these two distributions ($p < 0.01$). The plot reveals that the quartiles for the Lample et al. system are above those of the Ma and Hovy system. Using a Brown-Forsythe test, the standard deviations for the two distributions are different with $p < 0.05$. Table 5.2 shows the minimum, the maximum, and the median performance for the test performances.

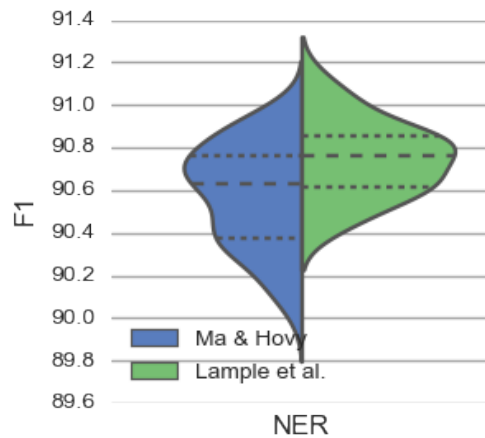


Figure 5.4: Distribution of scores for re-running the system by Ma and Hovy (left) and Lample et al. (right) multiple times with different seed values. Dashed lines indicate quartiles.

System	Reported F_1	# Seed values	Min. F_1	Median F_1	Max. F_1
Ma and Hovy	91.21%	86	89.99%	90.64%	91.00%
Lample et al.	90.94%	41	90.19%	90.81%	91.14%

Table 5.2: The system by Ma and Hovy (2016) and Lample et al. (2016) were run multiple times with different seed values.

Liu et al. (2017a) repeated our experiment and found similar variances for the two architectures. They further found that the performance from Ma and Hovy increases if it is trained on a GPU instead of a CPU. This difference between our scores and the reported scores from Ma and Hovy might be due to a difference between running the code on a CPU or a GPU.

In conclusion, training two non-deterministic approaches a single time and comparing their test performances is insufficient if we are interested to find out which approach is superior for a task. Large differences can occur due to better or worse sequences of random numbers.

In a usual setup, approaches are often trained multiple times, e.g., for tuning hyperparameters, and the model with the highest development score would be used for labeling the test data, i.e., be used to report the test performance. For the Lample et al. system we observe a Spearman’s rank correlation between the development and the test score of $\rho = 0.229$. This indicates a weak correlation and that the performance on the development set is not a reliable indicator. Using the run with the best development score (94.44%) would yield a test performance of mere 90.31%. Using the second best run on the development set (94.28%) would yield state-of-the-art performance with 91.00%. This difference is statistically significant ($p < 0.002$).

Best Run Comparison

Non-deterministic approaches can produce weak as well as strong models as shown in the previous section. Instead of training those a single time, we might want to compare only the “best” model for each approach, i.e., the models that performed best on the development set. This idea is formalized in Evaluation 2.

Table 5.3 depicts the results of comparing only the models that performed best on the development set. For all tasks, we observe small Spearman’s rank correlation ρ between the development and the test score. The low correlation indicates that a run with high development score does not have to yield a high test score.

Task	ρ	τ	% significant	$\Delta_{95}^{(dev)}$	$\Delta_{95}^{(test)}$	$\Delta_{Max}^{(test)}$
ACE - Entities	0.153	0.65	24.86%	0.42	1.04	1.66
ACE - Events	0.241	1.97	29.08%	1.29	3.73	7.98
CoNLL - Chunking	0.262	0.20	15.84%	0.10	0.29	0.49
CoNLL - NER-En	0.234	0.42	21.72%	0.27	0.67	1.12
CoNLL - NER-De	0.422	0.78	25.68%	0.58	1.44	2.22
GermEval - NER-De	0.333	0.60	16.72%	0.48	0.90	1.63
TempEval - Events	-0.017	1.19	9.38%	0.74	1.41	2.57

Table 5.3: The same BiLSTM-CRF approach was evaluated twice under Evaluation 2. ρ is the Spearman’s rank correlation coefficient between the development and the test score. The threshold τ column depicts the average difference in percentage points F_1 -score for statistical significance with $0.04 < p < 0.05$. The % significant column depicts the ratio how often the difference between $A_*^{(j)}$ and $\tilde{A}_*^{(j)}$ is significant. Δ_{95} depicts the 95% percentile of differences between $A_*^{(j)}$ and $\tilde{A}_*^{(j)}$. $\Delta_{Max}^{(test)}$ shows the largest difference.

For the *ACE 2005 - Events* task, we observe a significant difference between $A_*^{(j)}$ and $\tilde{A}_*^{(j)}$ in 29.08% of the cases. We observe for this task that the difference in test score can be as large as 7.98 percentage points F_1 -score between $A_*^{(j)}$ and $\tilde{A}_*^{(j)}$.

As before, we observe that $\Delta_{95}^{(test)}$ is much larger than τ , i.e., test performances of A_* vary to a large degree, larger than the threshold τ for statistical significance.

The table also depicts $\Delta_{95}^{(dev)}$, the 95% percentile of differences in terms of development performance. We observe a large discrepancy between $\Delta_{95}^{(dev)}$ and $\Delta_{95}^{(test)}$: For the 1,000 rows, we were able to find models $A_*^{(j)}$ and $\tilde{A}_*^{(j)}$ that performed comparably on the development set. However, the performance differs largely on the test set.

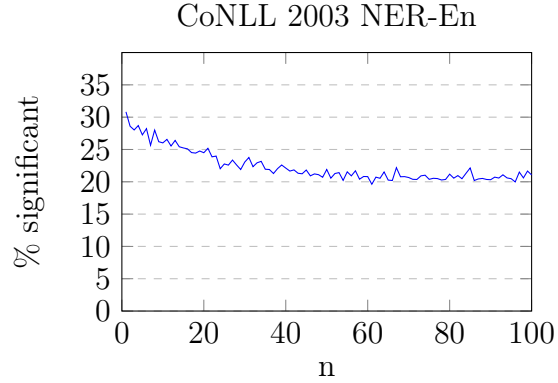


Figure 5.5: Ratio of statistically significant differences between A_* and \tilde{A}_* for different n -values.

We studied if the value of statistically significant differences between A_* and \tilde{A}_* depends on n , the number of sampled models. Figure 5.5 depicts the ratio for different n -values for the CoNLL 2003 NER-En task. We observe that the ratio of significant differences decreases with increasing number of sampled models n . However, the ratio stays flat after about 40 to 50 sampled models. For $n = 100$ we observe that 21.06% of the pairs are significant different with a $p < 0.05$ value.

5.4 Why Comparing Single Model Performances is Insufficient

While it is straightforward to understand why Evaluation 1 is improper for non-deterministic machine learning approaches, it is less obvious why this is also the case for Evaluation 2. If we ignore the bad models, where the approach did not converge to a good minimum, why can't we evaluate the best achievable performances of approaches?

The issue is not the significance test but has to do with the wrong conclusions we draw from a significant difference. The null-hypothesis for, e.g., the bootstrap test, is that two compared models would perform not differently on the complete data distribution. However, it is wrong to conclude from this that one approach is capable of producing better models than the other approach. The issue is that selecting a model with high test / true performance is only possible to a certain degree and the uncertainty depends on the development set.

We write the (hypothetical) performance on the complete data distribution as $\Psi^{(true)}$. The development and test score are finite approximations of this true performance

of a model.

We can rewrite the development score as $\Psi^{(dev)} = \Psi^{(true)} + \mathcal{X}^{(dev)}$ and the test score as $\Psi^{(test)} = \Psi^{(true)} + \mathcal{X}^{(test)}$. $\mathcal{X}^{(dev)}$ and $\mathcal{X}^{(test)}$ are two random variables with unknown means and variances stemming from the finite sizes of development and test set.

Given two models A_* and B_* , the significance test checks the null hypothesis whether $\Psi_{A_*}^{(true)}$ is equal to $\Psi_{B_*}^{(true)}$ given the two results on the test set.

When we select the models A_* and B_* based on their performance on the development set, we face the issue that the true performance is not monotone in the development score.

Assume we have models A_1 and A_2 with identical development performance. The development performance might be:

$$\begin{aligned}\Psi_{A_1}^{(dev)} &= \Psi_{A_1}^{(true)} + \mathcal{X}_{A_1}^{(dev)} = 80\% - 2\% = 78\% \\ \Psi_{A_2}^{(dev)} &= \Psi_{A_2}^{(true)} + \mathcal{X}_{A_2}^{(dev)} = 76\% + 2\% = 78\%\end{aligned}$$

The test performances might be:

$$\begin{aligned}\Psi_{A_1}^{(test)} &= 80\% + 1\% = 81\% \\ \Psi_{A_2}^{(test)} &= 76\% - 1\% = 75\%\end{aligned}$$

We compare this against model B_* from approach B , which as a test performance of $\Psi_{B_*}^{(test)} = 78\%$:

If we select A_1 for the comparison against B_* , the significance test might correctly identify that A_1 has a significantly higher test performance than B_* . However, if we select model A_2 , the significance test might identify that A_2 has a significantly lower test performance than B_* . As we do not know which model, A_1 or A_2 , to select for Evaluation 2, the outcome of Evaluation 2 is up to chance. If we select A_1 , we might conclude that approach A is better than approach B , if we select A_2 , we might conclude the opposite.

This situation becomes especially present for badly selected development sets. Assume $\Psi^{(dev)}$ is purely random. The choice for A_* is then random, and the test score differences between A_* and \tilde{A}_* from the same approach can become arbitrarily large. Often, less attention is paid to the selection of the development set, and in some cases, the development set is significantly smaller than the test set. However, we can select a model with high test score only to a certain degree, and this factor depends on how good the development score can predict the test or true performance of a model.

Distribution of $\Psi_{A_1}^{(test)} - \Psi_{A_2}^{(test)}$

We are interested to which degree test scores vary for two models with identical development scores.

We can write the scores as:

$$\begin{aligned}\Psi_{A_1}^{(dev)} &= \Psi_{A_1}^{(true)} + \mathcal{X}_{A_1}^{(dev)} & \Psi_{A_1}^{(test)} &= \Psi_{A_1}^{(true)} + \mathcal{X}_{A_1}^{(test)} \\ \Psi_{A_2}^{(dev)} &= \Psi_{A_2}^{(true)} + \mathcal{X}_{A_2}^{(dev)} & \Psi_{A_2}^{(test)} &= \Psi_{A_2}^{(true)} + \mathcal{X}_{A_2}^{(test)}\end{aligned}$$

We assume $\Psi_{A_1}^{(dev)} = \Psi_{A_2}^{(dev)}$, hence:

$$\begin{aligned}\Psi_{A_1}^{(true)} + \mathcal{X}_{A_1}^{(dev)} &= \Psi_{A_2}^{(true)} + \mathcal{X}_{A_2}^{(dev)} \\ \Rightarrow \Psi_{A_1}^{(true)} - \Psi_{A_2}^{(true)} &= \mathcal{X}_{A_2}^{(dev)} - \mathcal{X}_{A_1}^{(dev)}\end{aligned}$$

For the test performance difference, this leads to:

$$\begin{aligned}\Psi_{A_1}^{(test)} - \Psi_{A_2}^{(test)} &= (\Psi_{A_1}^{(true)} - \Psi_{A_2}^{(true)}) + (\mathcal{X}_{A_1}^{(test)} - \mathcal{X}_{A_2}^{(test)}) \\ &= (\mathcal{X}_{A_2}^{(dev)} - \mathcal{X}_{A_1}^{(dev)}) + (\mathcal{X}_{A_1}^{(test)} - \mathcal{X}_{A_2}^{(test)})\end{aligned}$$

The difference in test performance between A_1 and A_2 does not only depend on $\mathcal{X}^{(test)}$, but also on the random variable of the development set $\mathcal{X}^{(dev)}$. Hence, the variance introduced by the finite approximation of the development set is important to understand the variance of test scores. Significance tests, like the bootstrap test or the approximate randomized test, only take the variance by the finite sample size of the test set into account ($\mathcal{X}^{(test)}$). As shown, it is also important to take the variance introduced by the finite sample size of the development set ($\mathcal{X}^{(dev)}$) into account if we want to identify superior learning approaches.

Empirical Estimation

In this section we study how large the test score can vary for two models with identical development scores. We assume $\Psi_{A_1}^{(dev)} = \Psi_{A_2}^{(dev)}$. We are interested in how much the test score for these two models can vary, i.e., how large the difference $|\Psi_{A_1}^{(test)} - \Psi_{A_2}^{(test)}|$ can reasonably become.

We do this by computing a linear regression $f(\Psi^{(dev)}) \approx \Psi^{(test)}$ between the development and test score. For this linear regression, we compute the prediction interval ζ (Faraway, 2002). The test score should be within the range $f(\Psi^{(dev)}) \pm \zeta(\Psi^{(dev)})$ with a confidence of α .

The prediction interval is given by:

$$\zeta(\Psi^{(dev)}) = t_{n-2}^* s_y \sqrt{1 + \frac{1}{n} + \frac{(\Psi^{(dev)} - \overline{\Psi^{(dev)}})^2}{(n-1)s_x^2}}$$

with n the number of samples, t_{n-2}^* the value for the two-tailed t-distribution at the desired confidence α for the value $n-2$, s_y the standard deviation of the residuals calculated as:

$$s_y = \sqrt{\frac{\sum (\Psi^{(test)} - \hat{\Psi}^{(test)})^2}{n-2}}$$

$\overline{\Psi^{(dev)}}$ the mean value $\Psi_i^{(dev)}$ and s_x the unbiased estimation of standard deviation:

$$s_x^2 = \frac{1}{n-1} \sum_{i=1}^n \left(\Psi_i^{(dev)} - \overline{\Psi^{(dev)}} \right)^2.$$

An extreme difference in test score would be $\Psi_{A_1}^{(test)} \leq f(\Psi^{(dev)}) - \zeta(\Psi^{(dev)})$ for the one model and $\Psi_{A_2}^{(test)} \geq f(\Psi^{(dev)}) + \zeta(\Psi^{(dev)})$ for the other model. The difference would then be $|\Psi_{A_1}^{(test)} - \Psi_{A_2}^{(test)}| \geq 2\zeta(\Psi^{(dev)})$.

The probability of $|\Psi_{A_1}^{(test)} - \Psi_{A_2}^{(test)}| \geq 2\zeta(\Psi^{(dev)})$ is $(1-\alpha)^2$. We set $(1-\alpha)^2 = 0.05$. In this case, $|\Psi_{A_1}^{(test)} - \Psi_{A_2}^{(test)}| \leq 2\zeta(\Psi^{(dev)})$ in 95% of the cases.

The value of $2\zeta(\Psi^{(dev)})$ is approximately constant in terms of the development score $\Psi^{(dev)}$. Hence, we computed the mean $2\zeta(\overline{\Psi^{(dev)}})$ and depict the value in Table 5.4.

Task	Predict. Interval
ACE 2005 - Entities	1.03
ACE 2005 - Events	3.68
CoNLL 2000 - Chunking	0.25
CoNLL 2003 - NER-En	0.69
CoNLL 2003 - NER-De	1.24
GermEval 2014 - NER-De	0.88
TempEval-3 - Events	1.30

Table 5.4: Size of the 95% interval for the test scores of two models with the same development score.

The value 3.68 for the *ACE 2005 - Events* tasks indicates that, given two models with the same performance on the development set, the test performance can vary up to 3.68 percentage points F_1 -score (95% interval). The values $2\zeta(\overline{\Psi^{(dev)}})$ are comparably similar to the value of $\Delta_{95}^{(test)}$ in Table 5.3.

Evaluation 2 is interesting if we want to compare two methods for the best achievable test performances. However, identifying which regions in weight space will yield high test performance is difficult, and the development set introduces an uncertainty factor. The computed prediction interval could give a hint, which improvement in test score is needed to be certain that an approach can produce better working models than another approach for a specific dataset. Note, the estimated prediction intervals are much higher than what is typically observed as a significant improvement for these tasks.

5.5 Sources of Variation

Comparing two or more learning approaches based on the test performance of individual models can be misleading and bears a high risk of drawing wrong conclu-

sions. It is possible that observed differences are due to chance and are not due to a better learning approach. Deciding when an approach is superior for a task is non-trivial.

It is important to note different sources of variations that influence the performance of approaches. The first source of variance is the internal randomness of the learning approach. The performance of non-deterministic learning approaches can vary heavily depending on the sequence of random numbers. The second source is the selection of the training, development and test set. On one split, one approach might outperform the other, while on another split, the other approach is superior. The third source of variance can be label noise, e.g., from mislabeled and ambiguous samples.

5.5.1 Internal Randomness

The model performance for non-deterministic learning approaches that use a sequence of random numbers to train a model can depend heavily on this sequence. Some sequences lead to well-performing models, while other sequences lead to less well-performing models. Neural networks are an example of an approach that uses randomness in various parts. Typically, the weights are initialized randomly, training data is shuffled for stochastic gradient descent, and random dropout masks are applied to avoid overfitting.

The sequence of random numbers influences to which local minimum or saddle point the network converges. While the different convergence points in terms of the value of the cost function on the training set are often similar (Rumelhart et al., 1986; Keskar et al., 2016), and it is believed that local minima are not too different from the global minimum (Rumelhart et al., 1986; Kawaguchi, 2016), different convergence points show large differences in terms of generalizing to new, unseen data.

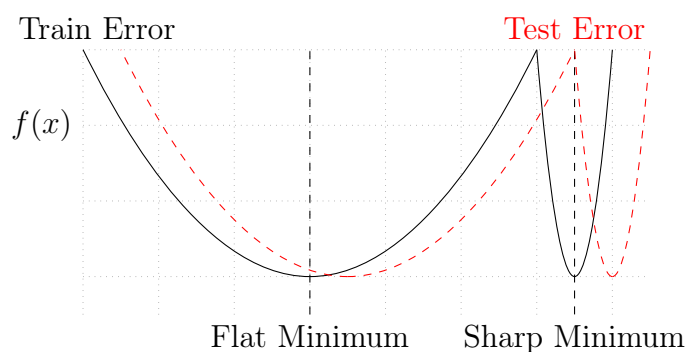


Figure 5.6: A conceptual sketch of flat and sharp minima from Keskar et al. (2016). The Y-axis indicates values of the error function and the X-axis the weight-space.

As (informally) defined by Hochreiter and Schmidhuber (1997b), a minimum can be flat, where the error function remains approximately constant for a large connected region in weight-space, or it can be sharp, where the error function increases rapidly in a small neighborhood of the minimum. A conceptual sketch is given in

Figure 5.6. The error functions for training and testing are typically not perfectly synced, i.e., the local minima on the train or development set are not the local minima for the held-out test set. A sharp minimum usually depicts poorer generalization capabilities, as a slight variation results in a rapid increase of the error function. On the other hand, flat minima generalize better on new data (Keskar et al., 2016). Keskar et al. observe for the MNIST, TIMIT, and CIFAR datasets, that the generalization gap is not due to *overfitting* or *overtraining*, but due to different generalization capabilities of the local minima, the networks converge to. To which type a neural network converges can be influenced, e.g., by the design of the network or by the training method. Keskar et al. (2016) observe that training with large mini-batches tends to converge to sharp minima. Hochreiter and Schmidhuber (1997b) and Chaudhari et al. (2016) propose alternative optimization algorithms that avoid sharp valleys.

Taking the internal randomness into account when comparing two learning approaches is crucial as shown in section 5.3. Otherwise, it is uncertain whether an improved model performance stems from a superior learning approach or if it stems from a luckier sequence of random numbers. Accounting for this source of variation in an evaluation is straightforward: Approaches are trained multiple times with varying sequences of random numbers and the resulting scores are compared.

5.5.2 Train, Development and Test Samples

For most tasks, only a finite number of samples are annotated. Those are usually split into a training, development and test set. A model might outperform another model on a particular test dataset, however, on the complete data population, both models would perform identically. This issue is typically addressed by performing a significance test that is based on the test scores of the models. Possible significance tests are the approximate randomized test (Riezler and Maxwell, 2005) or the bootstrap test (Berg-Kirkpatrick et al., 2012). The significance tests check the null-hypothesis that two models would perform identically on the whole (infinite) data population.

However, not only the finite size of the test dataset influences the outcome. The finite size of the training and development datasets are also sources of variations. Small changes to the training dataset, like adding or removing individual data points, may cause large differences in the model produced by a learning approach. Breiman (1996a,b) showed that this is a serious issue for decision tree algorithms. In Reimers and Gurevych (2015) we observed a similar case for the 2015 NIST TAC KBP Event Detection shared task: Using only 75% of the available training data, the performance of the proposed system improved by about 3 percentage points F_1 -score.

Training a neural network until the error function on the training set converges can lead to overfitting, i.e., it can lead to a model that generalizes badly to new data. Here, the development set plays a crucial role to determine when to stop the training process. Further, the development set is often used for parameter tuning. However,

this introduces another source of variation. Maybe an approach produced a well-performing model for the whole data distribution. But this good model performed rather badly on the development set and is discarded for the final selection. Instead, we select another model that performed well on the development set, but rather bad for the whole data distribution. We show in [section 5.3](#) and [section 5.4](#) that the correlation between the development score and the test score can be low. If we would train the approach with a different development set, we might select a different model that performs better for the whole data population.

Both, the variance introduced by the training dataset and by the development dataset are not factored into significance tests that work on the test output of a model, for example, the approximate randomized test ([Riezler and Maxwell, 2005](#)), the bootstrap test ([Berg-Kirkpatrick et al., 2012](#)), or the McNemar’s test ([Everitt, 1977](#); [Dietterich, 1998](#)). An approach might produce better working models for one particular partitioning of the dataset, however, for another partitioning, opposite conclusions might be drawn.

Instead of a fixed train, development and test set, cross-validation or other resampling methods could take the variation introduced by different partitioning into account. However, creating these different partitionings is not straightforward. For example, the CoNLL 2003 NER dataset and the ACE 2005 dataset splits documents based on the document creation date. Documents before a certain date are for training and development and documents after a certain date are used for testing. This temporal split tries to address the challenge that language changes over time.

A further issue is that cross-validation shouldn’t be used for tuning of algorithms. [Varma and Simon \(2006\)](#) showed that, when an algorithm is tuned using cross-validation, it can introduce a positive bias. The cross-validation performance is a too optimistic estimation of the performance on unseen data.

5.5.3 Random and Not So Random Class Noise

A certain fraction of the data is mislabeled, e.g., due to human errors from the annotators. Another fraction of the data is ambiguous, i.e., multiple labels for the same sample are plausible. However, most tasks allow only one label per sample.

The inter-annotator agreement can be seen as an estimate of these two fractions. The fraction of mislabeled and ambiguous samples depends on the difficulty of the task, on how the annotation was performed, and on the skill level of the annotators. The estimated error rate for the Penn Treebank POS dataset is approximately 3% ([Marcus et al., 1993](#)). The agreement on the span of events for the TimeBank corpus is estimated at 0.78 F_1 -score (cf. [Table 2.4](#)).

If labels for these two classes are assigned randomly, then no classifier can achieve an error rate less than the noise in the data. However, the assigned label is not necessarily random. It can be influenced by the annotation process, e.g., by proposing a default label for a sample, or it can be influenced by individual biases and understandings from annotators. Further, the understanding of the task can change over

time. Hence, samples annotated at the start are different from samples annotated at the end.

Annotation projects are rarely performed by a single annotator. Usually, the work is split between multiple annotators to enable the creation of larger datasets. Each annotator has a different understanding of the task, which can lead to different biases for the set of mislabeled and ambiguous samples. This set of samples is not necessarily equally distributed across data partitions, e.g., training data might be annotated mainly by one group of annotators, while test data might be annotated mainly by a different group. Hence, we can observe different biases in the training and in the test set.

Depending on the dataset, the fraction of mislabeled and ambiguous samples can be quite large, and it can influence the achieved performances for models. A model might not necessarily perform better on the test dataset, it just might have been lucky and guessed better the bias for mislabeled and ambiguous samples in the test set.

We observed this for the 2015 NIST TAC KBP Event Detection shared task (Reimers and Gurevych, 2015). The annotation of contact events was especially problematic for this dataset (cf. section 2.3.2). Contact events have a low inter-annotator agreement, and there is a high difference in the distribution of contact events in the train and in the test datasets. The guidelines state that for speech verbs like *said* or *told* only the first occurrence in a document is tagged. However, annotators varied in the implementation of this rule (Song et al., 2016). It appears only some annotators followed this rule. Our approach that was trained on 75% of the training data predicted by accident correctly how those contact events are labeled in the unseen test data. This accident improved the test performance by 3% percentage points F_1 -score. When we trained the approach on the whole training set, it guessed wrongly if and how annotators for the test data will implement this rule for contact events.

This class noise can make the comparison of approaches difficult. It might be that one approach is not better than another approach for a specific task, it just might be that one approach predicted the class noise in the test set better than the other approach.

5.6 Evaluation Methodologies Based on Score Distributions

We define the performance for a model as:

$$\Psi_{A(\text{Train}, \text{Dev}, \text{Rnd})}^{(\text{Test})} = S(A_{(\text{Train}, \text{Dev}, \text{Rnd})}(\text{Test}_x), \text{Test}_y). \quad (5.1)$$

A is the learning approach that trains a model given a training set **Train**, a development set **Dev** and a sequence of random numbers **Rnd**. The resulting model

$A_{(\text{Train}, \text{Dev}, \text{Rnd})}$ is applied to the test dataset Test_x and a performance score S is computed between the predictions and the gold labels Test_y .

Based on this, we define two idealized definitions for *approach A superior to approach B*.

Evaluation 3. Given a certain task and a potentially infinite data population \mathcal{D} . We call *approach A superior to approach B* for this task with training set of size $k \leq |\text{Train}| \leq l$ if and only if the expected test score for approach A is larger than the expected test score for approach B :

$$E \left[\Psi_{A(\text{Train}, \text{Dev}, \text{Rnd})}^{(\text{Test})} \right] > E \left[\Psi_{B(\text{Train}, \text{Dev}, \text{Rnd})}^{(\text{Test})} \right]$$

with Train , Dev , and Test sampled from \mathcal{D} .

We can approximate the expected test score for an approach by training multiple models and comparing the sample mean values $\overline{\Psi_{A_{1..n}}^{(\text{Test})}}$ and $\overline{\Psi_{B_{1..m}}^{(\text{Test})}}$. We conclude that one approach is superior if the difference between the means is significant.

A common significance test used in literature is the Welch's t-test. This is a simple significance test which only requires the information on the sample mean, sample variance and sample size. However, the test assumes that the two distributions are approximately normally distributed.

Evaluation 3 computes the expected test score, however, *superior* can also be interpreted as a higher probability to produce a better working model.

Evaluation 4. Given a certain task and a potentially infinite data population \mathcal{D} . We call *approach A superior to approach B* for this task with training set of size $k \leq |\text{Train}| \leq l$ if and only if the probability for approach A is higher to produce a better working model than it is for approach B . We call approach A superior to approach B if and only if:

$$P \left(\Psi_{A(\text{Train}, \text{Dev}, \text{Rnd})}^{(\text{Test})} \geq \Psi_{B(\text{Train}, \text{Dev}, \text{Rnd})}^{(\text{Test})} \right) > 0.5$$

We can estimate if the probability is significantly different from 0.5 by sampling a sufficiently large number of models from approach A and approach B and then applying either a Mann-Whitney U test for independent pairs or a Wilcoxon signed-rank test for matched (dependent) pairs for the achieved test scores.

In contrast to the Welch's t-test, those two tests do not assume a normal distribution. However, they ignore the difference between the performance scores and they could be excessively conservative.

Note that there is a fine distinction between Evaluation 3 and Evaluation 4. Evaluation 3 compares the mean values for two approaches, while Evaluation 4 compares the medians of the distributions.⁴ For skewed distributions, the median is different from the mean, which might change the drawn conclusion from Evaluation 3 and

⁴ Note, for certain distributions, the median m with $P(X \leq m) \leq 0.5$ and $P(X \geq m) \leq 0.5$ might not be uniquely defined. This does not affect Evaluation 4.

Evaluation 4. Approach A might have a better mean score than approach B , but a lower median than approach B or vice versa. If mean and median are identical, then Evaluation 3 and Evaluation 4 are identical.

Note, **Train**, **Dev**, and **Test** in Evaluation 3 and 4 are random variables sampled from the (infinite) data population \mathcal{D} . This is an idealized formulation for comparing machine learning approaches as it assumes that new, independent datasets from \mathcal{D} can be sampled. However, for most tasks, it is not easily possible to sample new datasets. Instead, only a finite dataset is labeled that must be used for **Train**, **Dev**, and **Test**. This creates the risk that an approach might be superior for a specific dataset, however, for other train, development, or test sets, this might not be the case (cf. [section 5.5.2](#) and [section 5.5.3](#)).

Evaluation 3 and Evaluation 4 both mention that training sets are of size $k \leq |\text{Train}| \leq l$. Learning approaches can react differently to increasing or decreasing training set sizes, e.g., approach A might be better for larger training sets while approach B might be better for smaller training sets. When comparing approaches, it would be of interest to know the lower bound k and the upper bound l for approaches A and B . However, most evaluations check for practical reasons only one training set size, i.e., $k = l$.

Experiment

We tested if the introduced evaluation setups *Evaluation 3* and *Evaluation 4* can address for the internal randomness of learning approaches. We test this by checking if these methods can reliably detect that there is no difference between the approach A and \tilde{A} .

We re-use the data from [section 5.3](#) and compare 25 models from approach A ($A_1^{(j)}, \dots, A_{25}^{(j)}$) against 25 models from approach \tilde{A} ($\tilde{A}_1^{(j)}, \dots, \tilde{A}_{25}^{(j)}$). For Evaluation 3, we use the Welch’s t-test, for Evaluation 4, we use the Wilcoxon signed-rank test. As threshold, we used $p < 0.05$.

Task	Eval. 3 (Mean Scores)	Eval. 4 (Median Scores)
ACE - Entities	4.68%	4.86%
ACE - Events	4.72%	4.67%
CoNLL - Chunking	4.60%	4.86%
CoNLL - NER-En	5.18%	5.01%
CoNLL - NER-De	4.83%	4.78%
GermEval - NER-De	4.91%	4.74%
TempEval - Events	4.72%	5.03%

Table 5.5: Percentage of significant difference between A and \tilde{A} for $p < 0.05$.

[Table 5.5](#) summarizes the outcome of this experiment. The ratios are all at about 5%, which is the number of false positives we would expect from a threshold $p < 0.05$. In contrast to Evaluation 1 and 2, Evaluation 3 and 4 were able to identify that the approaches are identical in most cases.

Figure 5.7 shows a boxplot for the average test score of the first n models of $A^{(j)}$ for the CoNLL 2003 NER-En task. For $n = 1$, i.e., only a single model with one sequence of random numbers is trained, we observe a large variance in test score. Some models perform well, while others perform less well. With increasing n , i.e., training multiple models and averaging the test scores, the variance in mean test scores decreases. For example, if we average the performance of 20 models, the average is rather stable at about 0.90 F_1 -score.

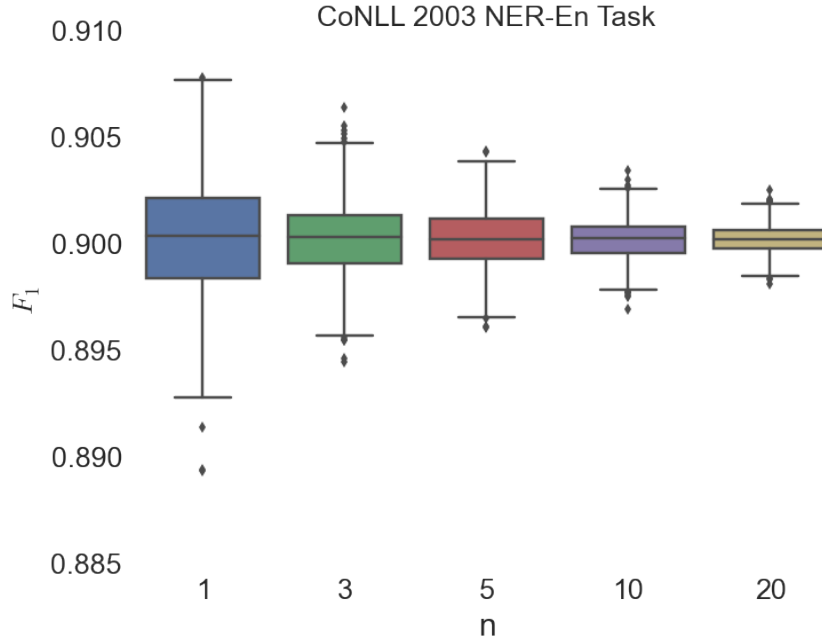


Figure 5.7: Averaging test-score of n models for the CoNLL 2003 NER task.

A large variance of the mean score $\overline{\Psi_{A_{1,\dots,n}^{(j)}}^{(test)}}$ will make it difficult to spot difference between two learning approaches. To express the variance in an intuitive value, we compute the 95th percentile $\Delta_{95}^{(test)}$ for the difference between the mean scores:

$$\Delta^{(test),(n,j)} = \left| \overline{\Psi_{A_{1,\dots,n}^{(j)}}^{(test)}} - \overline{\Psi_{\tilde{A}_{1,\dots,n}^{(j)}}^{(test)}} \right|$$

The value $\Delta_{95}^{(test)}$ gives an impression which improvement in mean test score is needed for a significant difference. Note, this value depends on the internal randomness of the learning approach and would be different for another learning approach.

The values are depicted in Table 5.6. For increasing n the value $\Delta_{95}^{(test)}$ decreases, i.e., the mean score becomes more stable. However, for the CoNLL 2003 NER-En task we still observe a difference of 0.26 percentage points F_1 -score between the mean scores for $n = 10$. For the ACE 2005 Events dataset, the value is even at 1.39 percentage points F_1 -score. Any refinement of our learning approach that does not result in a mean score improvement larger than the values depicted in Table 5.6 has a high risk to be insignificant.

Task	$\Delta_{95}^{(test)}$ for n scores				
	1	3	5	10	20
ACE - Entities	1.21	0.72	0.51	0.38	0.26
ACE - Events	4.32	2.41	1.93	1.39	0.97
CoNLL - Chunking	0.30	0.16	0.14	0.09	0.06
CoNLL - NER-En	0.83	0.45	0.35	0.26	0.18
CoNLL - NER-De	1.61	0.94	0.72	0.51	0.37
GermEval - NER-De	1.12	0.64	0.48	0.34	0.25
TempEval - Events	1.48	0.81	0.63	0.48	0.32

Table 5.6: 95% percentile of $\Delta^{(test)}$ after averaging the test performance for n models.

5.7 Hyperparameters

Hyperparameters play an important role in the performance of many learning approaches and can make the difference if a system is observed as *state-of-the-art* or as *mediocre* (Hutter et al., 2014). Tuning hyperparameters for an approach is non-trivial and Snoek et al. (2012) describe it as a “*black art that requires expert experience, unwritten rules of thumb, or sometimes brute-force search*”.

The difficult question arises how approaches should be compared with respect to the dependence on hyperparameters. Figure 5.8 shows the conceptual sketch of two approaches, for example, approach A a neural network with dropout (Srivastava et al., 2014b) and approach B the network without dropout, and their dependence on the number of hidden units. If only a single hyperparameter configuration is evaluated, all three conclusions are possible: Approach A outperforms approach B, both approaches perform equally well, and approach B outperforms approach A.

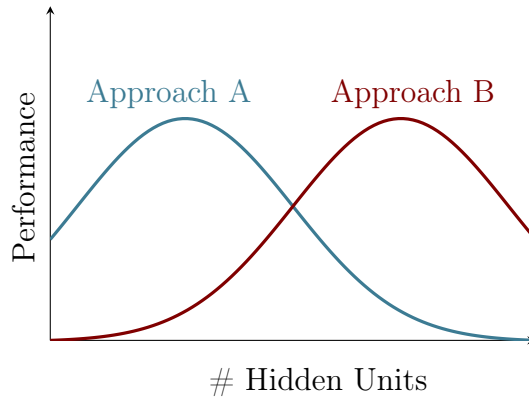


Figure 5.8: Hypothetical performance curve for two approaches depending on a single hyperparameter.

A fair comparison between approaches for a task requires that all approaches are tuned to the same degree. Otherwise, it cannot be differentiated if an approach is conceptually superior for a task or if it was only better tuned. Melis et al. (2017) demonstrated this issue when evaluating state-of-the-art neural networks for the

task of language modeling. With automatic black-box hyperparameter tuning, they conclude that standard LSTM architectures outperform more recent and supposedly better architectures.

However, ensuring that every compared approach is tuned to the same degree and with the same effort can be difficult. Often, tuning involves some manual steps, for example, selecting parameter boundaries. Here, the knowledge and experience of the researcher with that particular approach can influence the outcome of the tuning step. Even with a fully automated black-box hyperparameter tuning system, a fair comparison is not necessarily given. It might be that the tuning algorithm works better for the one approach than for the other approach. With another tuning algorithm, the opposite conclusion might be drawn.

Even when assuming a fair tuning for both approaches, deciding which approach is superior is conceptually complex. Figure 5.9 depicts a conceptual sketch of two approaches. Approach A is highly sensitive to the hyperparameters, and only a small range of parameters yield a good performance. In contrast, approach B is insensitive to the hyperparameters, and a large range of the hyperparameters gives a solid performance. However, the peak of approach B is not as high as it is for approach A.

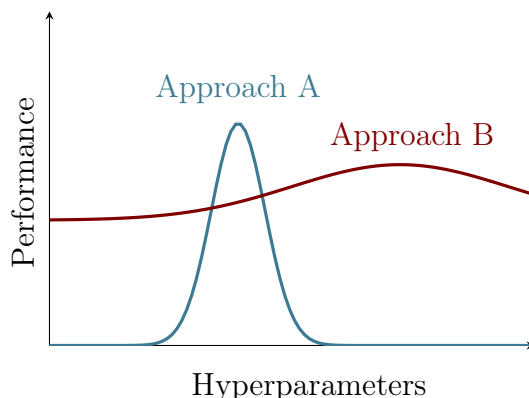


Figure 5.9: Approach A is sensitive to the hyperparameter, however, achieves a higher peak performance than approach B.

Deciding which approach from Figure 5.9 is superior is difficult. Training a deep neural network can be expensive and might require days to converge. In such a case, evaluating a large number of hyperparameters might be infeasible, especially as the parameters, that are important to tune, can change from task to task (Bergstra and Bengio, 2012). For such cases, approach B, that is less depended on the hyperparameters, might be the better choice. If training a model is cheap, more hyperparameters can be evaluated. In such a case, approach A might be preferable.

We observe these differences for actual datasets and architectures. For the Penn Treebank, we randomly sampled hyperparameters for a BiLSTM-architecture and trained it in a single task setup as well as in a multi-task setup using the CoNLL 2000 chunking dataset as an auxiliary task. Figure 5.10 depicts the violin plot of the achieved test accuracies. The violin plot is similar to a boxplot, however, it estimates the probability density function from the samples and depicts it along

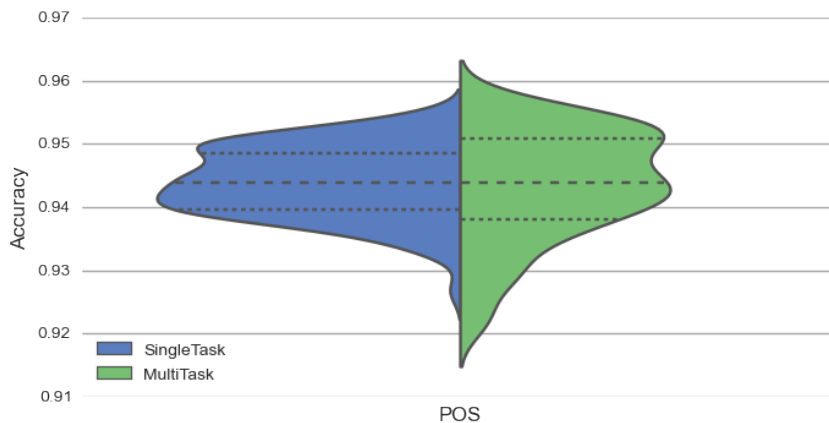


Figure 5.10: Comparison of the performance on the POS dataset. Blue/Left: Single Task Learning setup. Green/Right: Multi-Task Learning setup with Chunking as auxiliary data.

the Y-axis. If a violin plot is wide at a certain location, then achieving this test performance is especially likely. Besides the probability density, it also shows the median as well as the quartiles.

The violin plot reveals that the variance for the multi-task setup is larger, i.e., the chance of having an especially good but also an especially bad configuration is higher than for the single task setup. The median performance for both settings was identical. Further details on this experiment are presented in (Reimers and Gurevych, 2017b).

It remains an open question how hyperparameters should be properly taken into account when comparing different learning approaches. We might argue that the effort and time to tune a system should be neglected and only the performance from the best parameter setting matters. However, this might lead to approaches that are extremely sensitive to the selected hyperparameters and are difficult to apply for new tasks and datasets. Instead, we argue that the adaptation of approaches to new tasks is a critical aspect of an approach and the dependence on hyperparameters should be evaluated. We think the following aspects would be of interest:

- **Sensitivity:** How much does a slight change of the hyperparameter value affect the performance? Approaches that are less sensitive to concrete hyperparameter values are desirable, as those are potentially easier to adapt to new tasks. The sensitivity can be studied by training the approach multiple times with different values for one parameter. If the performance changes only slightly, we can conclude that the approach is less dependent on the value of that specific parameter. In chapter 3, we provide an evaluation for the BiLSTM-CRF architecture.
- **Importance:** For many approaches, it is unknown which parameters are important to tune. This potentially slows down the adaptation of the approach to new tasks. The importance can be studied, by training the approach for multiple tasks and comparing the optimal values for a parameter for each task.

If the optimal parameter value is comparably constant for all tasks, we would expect that this value will work well for new tasks.

- **Relative Importance:** Which parameters are important to tune can depend on the task. For one task, certain parameters might highly influence the achieved performance, while for another task, the same parameter impacts the performance negligibly. [Bergstra and Bengio \(2012\)](#) demonstrated this for various image recognition datasets. The relative importance can be studied by comparing how much the performance varies (sensitivity) for different tasks. If we observe that the set of parameters with high / low sensitivity changes, we can conclude that the relative importance of parameters depends on the task.
- **Effort of Tuning:** The perfect learning approach would be easy to adapt to new tasks, i.e., we would like to achieve a good performance with no or minimal tuning. We can estimate the effort needed for tuning by applying a hyperparameter optimization strategy and measuring the required time to find a configuration that performs within a threshold to the optimal configuration. For random hyperparameter search ([Bergstra and Bengio, 2012](#)), we could sample and evaluate n configurations and measure how many of these perform within a certain threshold. If most configurations perform nearly as good as the optimal configuration, we conclude that less effort must be spent on tuning the approach.

Future research requires the formalization of these four aspects, especially to make comparative evaluations of approaches possible.

5.8 Conclusion

Comparing machine learning approaches and finding the best, most accurate approach for a task is difficult. A common evaluation setup in the NLP community to find the *state-of-the-art technology* is based on shared tasks: An annotated dataset is partitioned into a training, development, and test set. Approaches can be trained and tuned on the training and development set, and a final model is selected. This final model is then evaluated on the unseen test data. The evaluation setups from previous shared tasks are also often used for future research, e.g., the dataset of the CoNLL 2003 shared task on NER is still actively used after 15 years.

The approach that produced the (significantly) best model in the shared task is then seen as the best approach for that task. However, as shown in this chapter ([section 5.2 - 5.3](#)), there is a high risk that this conclusion is wrong.

Non-deterministic approaches like neural networks can produce models with varying performances. If two approaches are trained only once, then it is not possible to decide whether the difference in performance stems from a better learning approach or from a luckier sequence of random numbers. We showed for the ACE 2005 dataset on events that the evaluated BiLSTM-CRF architecture can vary up to 9.04 percentage points F_1 -score if the same approach with the same configuration is

trained with two different sequences of random numbers (cf. Table 5.1). This does not only affect our implementation, but also the BiLSTM-CRF implementations from Ma and Hovy (2016) and Lample et al. (2016) (cf. Table 5.2).

In most shared tasks and for most research publications, approaches are not trained only once. Instead, approaches are tuned and many different models are created and evaluated on a development set. The best model on the development set is selected for the final evaluation.

As shown in section 5.3, when we compare two identical approaches, we see in a higher number of cases statistically significant difference in test performance. By implication, a significant difference in test performance does not allow to draw the conclusion that one approach is more accurate than the other. As before, the difference might be due to a luckier sequence of random numbers.

We generalized this observation in section 5.4 to any task and any approach that can produce models with varying performances. An interesting observation in that section is that the variance of the test scores depends on the development set. With an improper development set, the achieved test scores for the same approach can vary arbitrarily large. Without a good development set, we face the challenge of not knowing which configuration in weight space to choose for the final evaluation.

We conclude that the meaningfulness of a test score is limited by the quality of the development set. This is an important observation, as in many cases only little attention is paid to the selection of the development set. In order to have as much training data as possible, we often prefer small development sets, sometimes substantially smaller than the test set. Such a small development set increases the variance for the observed test scores and detecting differences between learning approaches becomes more challenging.

Further, significance tests like the approximate randomized test (Riezler and Maxwell, 2005), the bootstrap test (Berg-Kirkpatrick et al., 2012) or McNemar’s test (Everitt, 1977; Dietterich, 1998) do not take into account the uncertainty introduced by the development set. Hence, those tests are not suitable to compare learning approaches.

Section 5.5 identified three sources of variation for learning approaches: The internal randomness of the approach, the data partitioning into train, development and test set, and label noise from mislabeled or ambiguous samples. Accounting for the internal randomness in an evaluation is simple as shown in section 5.6: The approaches are trained multiple times with changing random sequences and either the mean scores or the score distributions are compared. Significance tests then indicate in most cases ($>95\%$ for $p < 0.05$) that there was no difference between the two identical learning approaches.

Accounting for variance introduced by data partitioning and from label noise is much more challenging and requires multiple, independently annotated datasets. Ideally, we would have multiple, independently annotated datasets for the same task that could be used for the evaluation of machine learning approaches. However, this is seldom the case. Instead, we propose to evaluate machine learning approaches on

many different tasks. If a new approach outperforms previous approaches on many tasks and datasets, it increases the confidence that this approach is actually superior and that the improvement is not due to a unique situation for one particular task with one particular dataset.

Chapter 6

Summary

In this thesis, we focused on three research questions in the area of event detection and extraction. In the following, we provide a summary of the conclusions for these three questions.

RQ1 Universal Learning Approach for Event Detection

Automatic event detection and extraction systems are an interesting research field with many different applications. However, how an event is defined and which linked information should be extracted from a text often depends on the concrete application. This led to many different annotation schemes, datasets, and systems, each having a slightly different definition what counts as an event and which further information is annotated / extracted.

Instead of a system that works well for one particular annotation scheme and dataset, we were interested to identify a universal learning approach for event detection. Event detection is often defined as a sequence tagging task. The BiLSTM-CRF architecture ([Huang et al., 2015](#); [Ma and Hovy, 2016](#); [Lample et al., 2016](#)) has been shown to work well for various sequence tagging tasks. However, it was unclear which of the various published design choices for this architecture are actually necessary to achieve a good performance. The various published extensions are difficult to compare from their original publications, as each author uses an own implementation with an own set of hyperparameters. Hence, it is unclear whether the improvement stems from the proposed extension or from, e.g., a better set of hyperparameters. This large number of design choices and hyperparameters poses the risk that time is wasted when adapting the approach to a new task by spending time on implementing and tuning unnecessary design choices or hyperparameters.

We implemented the various options and performed an evaluation that allows identifying which design choices and hyperparameters have an impact on the performance. We evaluated over 50,000 configurations on five common sequence tagging tasks. Our results show that only few options have a high impact on the performance, and the specific values for most hyperparameters are of minor relevance. The choices that had the highest impact were the pre-trained word embeddings and the usage of a CRF-classifier. We could observe average differences of up to 5 percentage points accuracy between different existent word embeddings. Embeddings that were

trained using dependency links worked better for syntactic-oriented tasks, and embeddings trained with context windows worked better for semantic-oriented tasks. This observation is in-line with previous observations, that dependency links capture better the functional properties of words (Levy and Goldberg, 2014; Komninos and Manandhar, 2016). Komninos and Manandhar presented a method to jointly train embeddings on dependency links and on context windows. The published embeddings by them achieved the best performance in all evaluated tasks.

Using the analysis, we developed a default configuration for the BiLSTM-CRF architecture and applied it to different event detection datasets. For all datasets, we observed high performance scores, comparable to specifically tuned approaches on those datasets. The only exception was the ACE 2005 dataset, where the BiLSTM-CRF architecture achieved a 7.3 percentage points lower F_1 -score. Recent system (Yang and Mitchell, 2016; Liu et al., 2017b) for the ACE 2005 dataset use joint training for detecting events and to extract event arguments. Event arguments can be critical for the distinction of the event type, for example, there is a difference if a person or an object is transported. Hence, joint training can improve the detection and classification of events. However, it requires that event arguments are annotated in the training set. The proposed BiLSTM-CRF architecture works without annotation of event arguments.

RQ2 Automatic Temporal Anchoring of Events

Events are strongly connected to the concept of time, and knowing when an event happened is crucial for a lot of applications. Hence, in chapter 4, we focused on the temporal anchoring of events.

We evaluated two widely used schemes to provide temporal information for events. The first scheme provides temporal information as an argument to an event and is, for example, used by the ACE and ERE standard. However, the scope is limited to noun phrases in the same sentence of the event. As a consequence, only 19.8% of the events in the ACE 2005 dataset have temporal information. The second scheme is based on temporal relations (TLINKs) between events and temporal expressions. An example for this approach is TimeML and the TimeBank (Pustejovsky et al., 2003).

The issue with TLINKs is that the number of links grows quadratic with the number of events and temporal expressions. An article with 200 events and temporal expressions would have 19,900 possible TLINKs. As annotating a large number of relations is infeasible, existent corpora restrict which TLINKs should be annotated. TimeBank annotated only salient links, however, the agreement which links are salient was rather low. Consecutive research usually focuses on only annotating TLINKs with the same sentence or between sentences. In our annotation study, we showed that in 58.7% of the cases the most informative temporal expression is more than one sentence apart from the event mention. For around 25% of the events, the most informative temporal expression is even five or more sentences away. Limiting the TLINKs to pairs that are at most one sentence apart poses the risk that important TLINKs are not annotated and consequently cannot be learned by automated systems.

We developed a new annotation scheme to provide precise temporal anchors for all events in a document. The annotators provide the exact event date if it is mentioned in the text. Otherwise, they provide a timeframe as precisely as possible when the event has happened. The annotators take the complete document into account to answer the question when the event happened. The annotation is not restricted to dates mentioned in the text. For example, in the sentence *It's the [second day]_{date:1998-03-06} of an [offensive]_{beginPoint=1998-03-05...}* it is clear that the offensive started on March 5th, 1998, even though this date is not explicitly stated in the document. In comparison to the TimeBank-Dense corpus (Cassidy et al., 2014), the temporal anchoring for events is of higher precision while requiring about 85% lower annotation efforts.

While the proposed scheme is simple for human annotators, it creates some challenges for automatic approaches. There is an infinite number of labels, and the label depends on the content of the document. Further, the complete document must be taken into account, and the human annotations do not provide the information which parts of the document are relevant for inferring the label. Hence, automatic systems must work on the complete document, must estimate if an event lasted longer than a day, and must decide whether the event time is explicitly stated or if it must be inferred from temporal ordering, causality, and world knowledge.

We propose a system for the developed annotation scheme. It is a decision tree that applies convolutional neural networks in its nodes. It performs multiple decisions and takes the complete document into account in order to produce the final event time label. On our annotated dataset, it performs substantially better than baselines based on state-of-the-art methods for TLINK extraction. We applied the model out-of-the-box for the task of automatic timeline generation for the SemEval-2015 Task 4 dataset. There, it achieved an improvement of 4 percentage points compared to the state-of-the-art approach for this dataset.

RQ3 Evaluation of Machine Learning Approaches

Our research community spent a lot of effort on identifying new learning approaches for tasks of interest. It is expected that new approaches are compared against existent approaches and that new approaches are in some way better, e.g., by achieving a higher performance. Wrong conclusions through insufficient evaluation methodologies can cause a lot of harm, e.g., by allocating researchers' time to try to reproduce those experiments.

In chapter 5 we described that there is a fine, but important distinction between *learning approach evaluation* and *model evaluation*. A *learning approach* describes the holistic setup to solve a certain optimization problem. For neural networks, this would be the network architecture, the optimization algorithm, the loss-function etc. A *model* is a specific configuration of the weights for this architecture. Such a model can be applied to new instances. In a specific application, where we want to use the model for labeling new data, we are interested to identify the best model for that task.

A common *model evaluation setup* is to train and tune the approach on a training and a development set. The model that performed best on the development set is selected

and is evaluated on unseen test data. A significance test like the approximate randomized test (Riezler and Maxwell, 2005), the bootstrap test (Berg-Kirkpatrick et al., 2012) or the McNemar’s test (Everitt, 1977; Dietterich, 1998) can be used to test if this model performs significantly better than previous models.

Instead of finding one particular weight configuration (i.e. a model) that works well for the test set, in research, we are often more interested to identify better learning approaches for that task. However, we observe that the described *model evaluation setup* is commonly used to show the improvement for a new learning approach. A new approach is trained, and when the resulting model achieves a statistically significant higher performance on the test set, the conclusion is drawn that this approach is superior to previous approaches. This form of evaluation is also used for shared tasks, where models from competing teams are evaluated. The usual goal for a shared task is not to identify one specific weight configuration that works well for the test dataset but to identify a superior learning approach for that task.

In chapter 5 we show that the conclusion *superior model performance for a task* \Rightarrow *superior learning approach for a task* cannot be drawn. We showed that when training two identical learning approaches, then there is a high chance to observe statistically significant performance differences. These differences are due to the internal randomness of learning approaches and due to uncertainty introduced by the development set: The best model on the development set must not be the best one on the test set. By implication, observing statistically significant model performance differences does not allow to conclude that the underlying learning approaches are different or that one approach is superior to the other. There is a high risk that these differences are due to chance.

This is an important observation for scientific publications and for the organization of shared tasks. If the goal is to compare approaches, it is insufficient to compare the performance of individual models. Instead, we propose the comparison of score distributions. We described three sources of variation, which can impact the performance: The internal randomness of the approach, the partitioning of the data, and class noise. Addressing the first source (the internal randomness) is straightforward: Multiple models are trained with different random sequences, and score distributions are compared. The two other sources are far more difficult to address. Addressing these sources would require that further labeled data is available. In conclusion, we recommend comparing new approaches always on multiple tasks and datasets. This reduces the risk that a new approach is only superior due to a special property of the dataset.

Future Research Directions

Each research question bears a lot of potential for future research. In the following, we list a subjective overview of the most crucial research directions for future work.

RQ1 Universal Learning Approach for Event Detection

- *Transfer learning:* The definition what counts as an event often depends on the specific application. This led to many different annotation schemes and datasets. So far, approaches were mainly trained and tuned in isolation on only one dataset. A transfer between datasets could have a lot of potential. Often, the event definitions share some similarities and only differentiate in details. For example, TimeML defines an event as a cover term for situations that happen or occur, and most verbs count as an event. In contrast, the ACE 2005 standard restricts the annotated events to 33 semantic classes, and other events, even though possible, are not annotated. A successful transfer learning approach could help to reduce the required training data for new definitions and tasks.
- *Extraction of New Event Argument Types:* In most applications, events must not only be detected, but also relevant information linked to the event must be extracted, for example, the participants or the place. As before, which arguments are considered as relevant is different for each application. Hence, each annotation scheme and dataset defines different event arguments. Extracting a new event argument type, that was not defined in the original scheme and dataset, is not possible with the existent approaches. A new event argument type typically requires the annotation of data and the training of the approaches from scratch. This prevents an easy adaptation of systems to new use-cases.
- *Event Importance:* The Oxford dictionary mentions that events are “*A thing that happens [...], especially one of importance.*” However, the notion of importance is not incorporated in most event detection datasets. Every annotated event, for example in a news article, has the same weight, even though some events are more relevant to the story. Estimating the importance of an event is critical for many applications, for example for story summarization or for creating news digests.

RQ2 Automatic Temporal Anchoring of Events

- *Temporal Anchoring of Complex Event Types:* Documents do not only report about events that are easily observable and that actually happened, but also about abstract, negative, future, hypothetical, conditional, uncertain, and generic events. Our annotation study showed that temporal anchoring of those events was especially challenging with low inter-annotator agreements. For example, when is the begin and end point of a conflict that did not happen? Even though the conflict did not happen, the author maybe refers to a certain time range when the event could have happened. A better understanding, how the human performs temporal inference and temporal anchoring for these types of events, is needed.
- *Temporal Anchoring for Heterogeneous Text Domains:* Most datasets on temporal anchoring of events and hence most automatic systems are based on news articles. In other textual domains, for example in novels, encyclopedic documents, research papers, or social media posts, temporal information is

expressed quite different to news articles. Further, events in a news article are often tightly connected to the document creation date as news articles usually report about recent events. Many systems heavily rely on this relation to predict when an event happened. For novels or encyclopedic documents, there is often no relation with the document creation time, and events can occur arbitrarily before or after the creation time.

RQ3 Evaluation of Machine Learning Approaches

- *Selection of Train, Development, and Test Set:* Nearly no systematic insights exist on how to create training, development, and test datasets. The annotated documents should be a representative sample of the (infinite) data distribution. However, it is unclear how to create a representative corpus. Further, it is unclear how to split the samples into a training, development, and test set.
- *Predictive Power of Performance Scores:* We use test scores to predict how well a model would perform on the complete, infinite data distribution. A model with a statistically significant higher test score is assumed to work better on the complete data distribution. However, when applying models in real-world applications, it can often be observed that the correlation between test scores and real-world performance is rather low. Some models and approaches are especially prone to changes in data distributions, dramatically lowering their performance in most real-world applications where the data distribution is different from the distribution in the train and validation set. Understanding why this occurs and identifying a method to address this in an evaluation setup could help to develop new approaches that are less sensitive to changing data distributions.
- *Comparison of Approaches:* Comparing a new approach with existent approaches is a difficult task and can easily lead to wrong conclusions. For example, the authors [Huang et al. \(2015\)](#), [Ma and Hovy \(2016\)](#), and [Lample et al. \(2016\)](#) developed each own implementations of the BiLSTM-CRF architecture, each with a different pre-processing and a different set of hyperparameters. They successively proposed new extensions to the BiLSTM-CRF architecture and showed performance increases on common datasets. However, it is unclear if the published performance improvements really stem from the proposed extensions. Small changes in the implementation, in the pre-processing, in the hyperparameters, in the software framework, or in the hardware can lead to large performance differences. For example, all three publications use different pre-trained word embeddings, and as shown in [section 3.5.1](#), the impact of the pre-trained word embeddings on the performance is extremely large.

In our experiment, we observed that the proposed extensions by [Ma and Hovy \(2016\)](#) and [Lample et al. \(2016\)](#) only led to small improvements for some datasets. For other datasets, we could not observe an improvement. A fair comparison of approaches requires that all other parameters, e.g., the pre-processing, the software framework, and the hardware, are identical and that all approaches are tuned to the same degree. Ensuring this is time-consuming, and more efficient methods to ensure a fair comparison are desirable.

Acknowledgments

I would like to express my gratitude towards Prof. Dr. Iryna Gurevych for giving me the opportunity to conduct this research, for her excellent supervision, and her valuable feedback. Further, I would like to thank Prof. Dr. Gerhard Weikum and Prof. Dr. Dan Roth for finding the time to review my thesis.

I thank my colleagues at the UKP Lab for their valuable constructive feedback that I received during various talks, discussions, and presentations. Also, I would like to express my gratitude towards my student research assistants Dushyanta Dhyani, Nazanin Dehghani, and Ekaterina Chernyak, and my thesis students Philip Beyer, Ziyang Li, Michael Bräunlein, Lucas Pradel, and Dominik Püllen for working with me on various projects and research questions. Further, I thank the ACL community for providing reviews for my publications and for giving me an opportunity to present and discuss my research at conferences with an international audience.

This work has been supported by the German Research Foundation as part of the Research Training Group Adaptive Preparation of Information from Heterogeneous Sources (AIPHES) under grant No. GRK 1994/1 and by the Volkswagen Foundation as part of the Lichtenberg-Professorship Program under grant No. I/82806. Additional support was provided by the German Federal Ministry of Education and Research (BMBF) as a part of the Software Campus program under the promotional reference 01-S12054. Calculations for this research were conducted on the Lichtenberg high performance computer of the TU Darmstadt.

Ehrenwörtliche Erklärung¹

Hiermit erkläre ich, die vorgelegte Arbeit zur Erlangung des akademischen Grades “Dr.-Ing.” mit dem Titel “Universal Machine Learning Methods for Detecting and Temporal Anchoring of Events” selbständig und ausschließlich unter Verwendung der angegebenen Hilfsmittel erstellt zu haben. Ich habe bisher noch keinen Promotionsversuch unternommen.

Darmstadt, den 14. Februar 2018

Nils Fabian Reimers

¹ Gemäß §9 Abs. 1 der Promotionsordnung der TU Darmstadt

Appendix

A Guidelines for Annotating Event Time Values

The following guidelines were used for the annotation of the event time (cf. [chapter 4](#)).

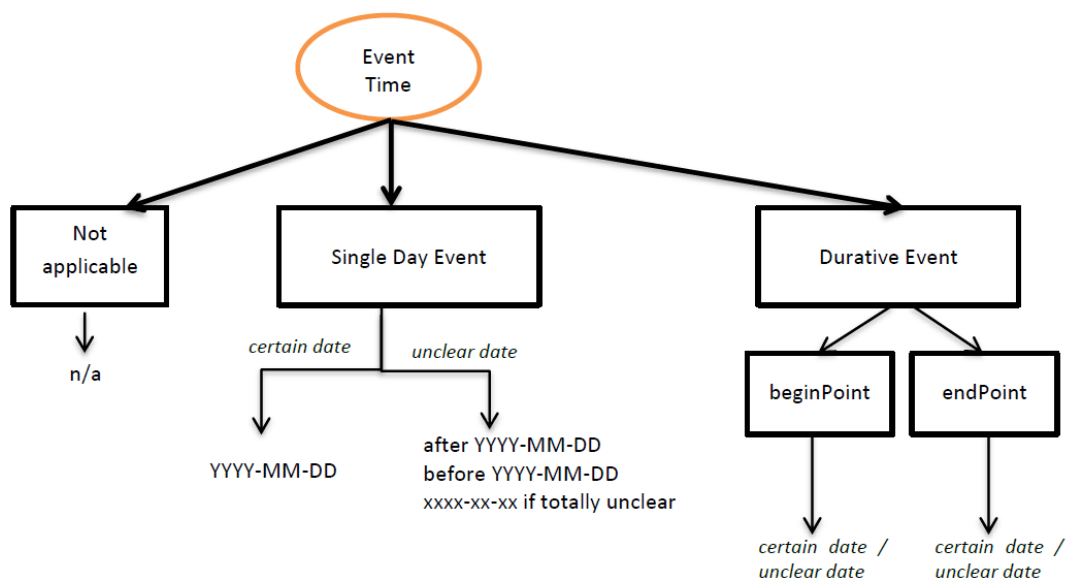


Figure A.1: Stepwise decision process for annotating event times.

1. **Decision:** Decide on the type of the event (single day event, durative, n/a)
 - **Timespan / durative:** The event or state occurs (or could have occurred) over several days (>1 day). Guiding questions: Given you would have complete knowledge of everything (on the past as well as on the future) and you would draw the duration of the event into a timeline, would it span over several days?
 - Annotate durative events always with beginPoint and endPoint
 - If something occurs repeatedly for an unknown number of times and with unknown time information, annotate as durative event with beginPoint when the event happened for the first time and endPoint when the event happened the last time.
 - **Singular Time Point:** Events that either occur instantaneously or only span over a single day.

- **Not applicable:** Annotate annotation errors and all events where a reasonable time information cannot be provided with n/a

Notes on Single Day Events

- Annotate events that occur instantaneously (like find, die, arrive, born) and events which span only over a single day (On Monday I [drove] Event to the store) as Single Day Events
- In case there is no clear date specified, use after and before to narrow down when the event could have happened.
- When you cannot decide whether the event happened before or after DCT, use xxxx-xx-xx
- For repeated events and you know the number and approximately the times of the events, annotate as date1, date2, date3 (e.g. the car [bombings] EventTime:1997-04-xx,1998-03-xx in April 1997 and March 1998 ...)
 - If the number and dates of the repeated events are unknown, annotate as durative event (e.g. Over the last 5 years, 100 people died in car [bombings] EventTime: beginPoint=1990-01-01 endPoint=DCT for a document with DCT=1995)

Simplifications for Single Day Events

- 2015-10-xx as simplification for after 2015-10-01 before 2015-10-31
- 2015-xx-xx as simplification for after 2015-01-01 before 2015-12-31
- You can leave the after or the before value blank if you cannot make a reasonable statement
- If something happened sometime during a season, annotate as after 2015-SP before 2015-SP

Document creation time and assumed dates

- Annotate date slots with *assumed DCT* when it is likely that the event occurred on the DCT time.
- When you know for sure it happened on DCT date, annotate as *DCT* (e.g. [Today]Time:DCT , they [announced]Event -> *EventTime: DCT*)
- For some news wire articles, like Wall Street Journal (WSJ), the DCT time is the publishing date of the article. The publishing date is one day after the article was written. There, use *assumed DCT-1* and *before DCT-1* / *after DCT-1* instead of *DCT*

Notes on Multi-Day Events

- Always annotate with both beginPoint and endPoint

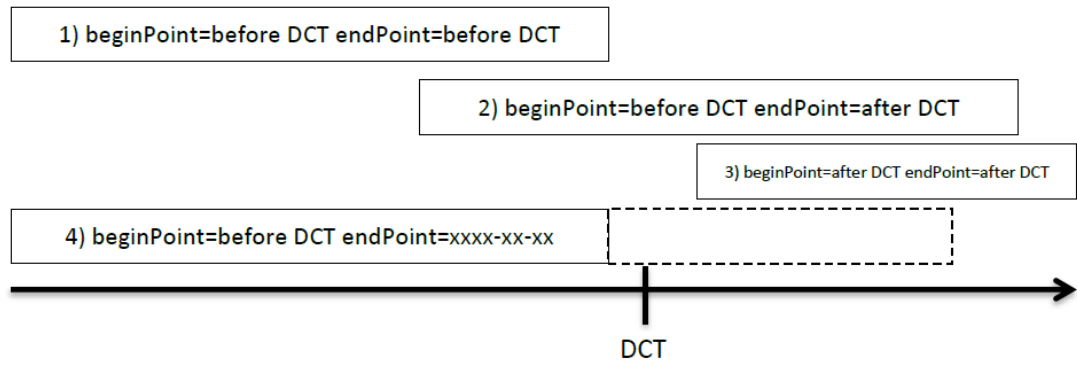


Figure A.2: Four different cases for the duration of a Multi-Day Event.

- There are 4 different cases (Figure A.2):
 1. Multi-Day event that started in the past and likely already ended
 2. Multi-Day event that started in the past and for sure did not yet end
 3. Multi-Day event that will start in the future
 4. Multi-Day event that started in the past and we don't know if it already ended at DCT time
- When there are more precise time information given, e.g. we know it started or ended on/before/after a certain date, put the more precise date into the beginPoint/endpoint
- If we only know the year & month of the begin date, e.g. 1998-03, write 'after 1998-03-01'. If we know the month of the end date, write 'before 1998-03-31'.
- If we only know the year, e.g. 1998, write 'beginPoint=after 1998-01-01 endPoint=before 1998-12-31'
- Properties and states are also considered durative events. If we only know that the property/state is true at DCT-time, annotate as 'beginPoint=before DCT endPoint=after DCT'. Example ("He owns 50% of the company" -> we can assume that he owned it before DCT and that he will own it after DCT)

Simplification for Multi-Day Events:

- 2015-Q1 as simplification for beginPoint=2015-01-01 endPoint=2015-03-31
- Spring: 2015-SP, Summer: 2015-SU, Fall: 2015-FA, Winter: 2015/2016-WI (Winter: beginPoint=2015-12-xx endpoint=2016-03-xx)
- Always annotate calendar quarters. Convert fiscal quarters to calendar quarters.

Notes on Generic Events

- Often we cannot decide precisely on the event time for generic or conditional events. When the articles tasks about past generic/conditional events, typically annotate those as before DCT.
- When the articles talks about future generic/conditional events, typically annotate those as after DCT Should there be more precise time information, use those.

List of Figures

2.1	Inter-annotator agreement for NIST TAC KBP 2015 events dataset. .	24
2.2	Inter-annotator agreement for NIST TAC KBP 2015 events dataset per type.	25
3.1	Illustration from Huang et al. (2015) on the proposed LSTM archi- tectures for sequence tagging.	33
3.2	Strategy from Lample et al. (2016) to derive character-based word representations.	34
3.3	Strategy from Ma and Hovy (2016) to derive character-based word representations.	34
3.4	Our BiLSTM-CRF architecture used for sequence tagging.	35
3.5	Probability density function using a softmax classifier or a CRF classifier	41
3.6	Performance on the CoNLL 2000 chunking shared task for various optimizers.	48
3.7	Performance on the CoNLL 2003 NER shared task for various opti- mizers.	48
3.8	Performance on the ACE 2005 entities dataset for various tagging schemes.	52
3.9	Plot of softmax and CRF classifier	54
3.10	Softmax and CRF classifier for different number of stacked layers . .	56
3.11	Number of recurrent units	60
3.12	Conceptual sketch of flat and sharp minima	64
4.1	Example of the proposed annotation scheme.	70
4.2	Comparison TimeBank and TimeBank-Dense	73
4.3	Distribution of distances in sentences between the event mention and the most informative temporal expression	81
4.4	Structure of the proposed system for event time extraction	86
4.5	Neural network architecture for local classifiers	86
5.1	Common evaluation methodology to compare two approaches for a specific task.	104
5.2	Single score comparison for non-deterministic learning approaches (Evaluation 1).	107
5.3	Illustration of model tuning and comparing the best models A_* and B_* (Evaluation 2).	108

LIST OF FIGURES

5.4	Distribution of scores for re-running the system by Ma and Hovy (left) and Lample et al. (right) multiple times with different seed values. Dashed lines indicate quartiles.	111
5.5	Ratio of statistically significant differences between A_* and \tilde{A}_* for different n -values.	113
5.6	Conceptual sketch of flat and sharp minima	117
5.7	Averaging test-score of n models for the CoNLL 2003 NER task. . . .	123
5.8	Hypothetical performance curve for two approaches depending on a single hyperparameter.	124
5.9	Approach A is sensitive to the hyperparameter, however, achieves a higher peak performance than approach B.	125
5.10	Single Task vs. Multi-Task Learning	126
A.1	Stepwise decision process for annotating event times.	139
A.2	Four different cases for the duration of a Multi-Day Event.	141

List of Tables

2.2	Properties of event corpora.	19
2.3	TimeBank statistics	20
2.4	Inter-annotator agreement for TimeBank	20
2.5	Inter-annotator agreement for TempEval-3 platinum corpus	21
2.6	Statistics for corpora that are based on TimeBank.	22
2.7	Statistics for the ACE 2005 corpus.	23
2.8	Statistics for the NIST TAC KBP 2015 events dataset.	24
2.9	Label distribution for ACE 2005 and TAC KBP 2015 dataset.	26
3.1	Overview of the benchmark tasks and number of sentences for the training, development and test set.	40
3.2	Comparison of CRF and softmax classifier	42
3.4	Comparison of Word Embeddings	46
3.5	Comparison of character-based word representations	47
3.6	Comparison of optimizers	49
3.7	Increasing learning rate for Adam optimizer	50
3.8	Comparison of gradient clipping	51
3.9	Comparison of gradient normalization	52
3.10	Comparison of tagging schemes	53
3.11	Comparison of softmax and CRF classifier	55
3.12	Comparison of dropout methods	57
3.13	Comparison of variational dropout	57
3.14	Comparison of the number of LSTM-layers	58
3.15	Optimal number of recurrent units	60
3.16	Comparison of mini-batch sizes	61
3.17	Recommended configuration for the BiLSTM-CRF architecture.	65
3.18	Overview of corpora used to evaluate the BiLSTM-CRF architecture for the task of event detection.	65
3.19	Evaluation of BiLSTM-CRF architecture on event detection	66
3.20	BiLSTM-CRF performance in comparison to state-of-the-art performances	67
4.1	Statistics for corpora that use TLINKs.	74
4.2	Statistics on the annotated event times	80
4.3	Comparison of dense TLINK annotation to the proposed scheme	82
4.4	System performance for automatic event time extraction.	94
4.5	Comparison of proposed system to state-of-the-art method	95

LIST OF TABLES

4.6	Comparison of proposed system to baselines	96
4.7	Accuracy of local classifiers	96
4.8	Percentage of labels in the test set affected by out-of-document dates.	97
4.9	Ablation test for the proposed event time extraction system.	97
4.10	Performance of the proposed system on the SemEval-2015 Task 4 Track B	99
5.1	Results for Evaluation Method 1	110
5.2	The system by Ma and Hovy (2016) and Lample et al. (2016) were run multiple times with different seed values.	111
5.3	Results for Evaluation Method 2	112
5.4	Size of the 95% interval for the test scores of two models with the same development score.	116
5.5	Percentage of significant difference between A and \tilde{A} for $p < 0.05$. . .	122
5.6	95% percentile of $\Delta^{(test)}$ after averaging the test performance for n models.	124

Bibliography

ACE (Automatic Content Extraction) English Annotation Guidelines for Events, 5.4.3 2005.07.01 edition, 2005.

Rodrigo Agerri, Eneko Agirre, Itziar Aldabe, Begoña Altuna, Zuhaitz Beloki, Egoitz Laparra, Maddalen López de Lacalle, German Rigau, Aitor Soroa, and Rubén Urizar: ‘NewsReader project’, in: *30th Conference of the Spanish Society for Natural Language Processing (SEPLN)*, 2014.

Jacqueline Aguilar, Charley Beller, Paul McNamee, Benjamin Van Durme, Stephanie Strassel, Zhiyi Song, and Joe Ellis: ‘A Comparison of the Events and Relations Across ACE, ERE, TAC-KBP, and FrameNet Annotation Standards’, in: *Proceedings of the Second Workshop on EVENTS: Definition, Detection, Coreference, and Representation*, pp. 45–53, Association for Computational Linguistics, Baltimore, Maryland, USA, June 2014.

James Allan: ‘Topic Detection and Tracking: Event-based Information Organization’, pp. 1–16, Kluwer Academic Publishers, Norwell, MA, USA, 2002.

Cosmin Bejan and Sanda Harabagiu: ‘Unsupervised Event Coreference Resolution with Rich Linguistic Features’, in: *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pp. 1412–1422, Association for Computational Linguistics, 2010.

Y. Bengio, P. Simard, and P. Frasconi: ‘Learning Long-term Dependencies with Gradient Descent is Difficult’, *Trans. Neur. Netw.* 5 (2): 157–166, March 1994.

Taylor Berg-Kirkpatrick, David Burkett, and Dan Klein: ‘An Empirical Investigation of Statistical Significance in NLP’, in: *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL ’12, pp. 995–1005, Association for Computational Linguistics, Stroudsburg, PA, USA, 2012.

James Bergstra and Yoshua Bengio: ‘Random Search for Hyper-parameter Optimization’, *Journal of Machine Learning Research* 13: 281–305, February 2012.

Christopher M. Bishop: *Pattern Recognition and Machine Learning*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

André Bittar, Pascal Amsili, Pascal Denis, and Laurence Danlos: ‘French TimeBank: An ISO-TimeML Annotated Reference Corpus’, in: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language*

- Technologies: Short Papers - Volume 2*, HLT '11, pp. 130–134, Association for Computational Linguistics, Stroudsburg, PA, USA, 2011.
- Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov: ‘Enriching Word Vectors with Subword Information’, *arXiv preprint arXiv:1607.04606* 2016.
- Brian Boyd: *On the Origin of Stories*, Harvard University Press, 2009.
- Philip Bramsen, Pawan Deshpande, Yoong Keok Lee, and Regina Barzilay: ‘Inducing Temporal Graphs’, in: *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, pp. 189–198, Association for Computational Linguistics, Stroudsburg, PA, USA, 2006.
- Leo Breiman: ‘Bagging Predictors’, *Mach. Learn.* 24 (2): 123–140, August 1996b.
- Leo Breiman: ‘Heuristics of instability and stabilization in model selection’, *Annals of Statistics* 24 (6): 2350–2383, 12 1996a.
- Taylor Cassidy, Bill McDowell, Nathanael Chambers, and Steven Bethard: ‘An Annotation Framework for Dense Event Ordering’, in: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 501–506, Association for Computational Linguistics, Baltimore, Maryland, USA, 2014.
- Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia: ‘SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Crosslingual Focused Evaluation’, in: *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pp. 1–14, Vancouver, Canada, 2017.
- Nathanael Chambers, Taylor Cassidy, Bill McDowell, and Steven Bethard: ‘Dense Event Ordering with a Multi-Pass Architecture’, *Transactions of the Association for Computational Linguistics* 2: 273–284, 2014.
- Nathanael Chambers and Dan Jurafsky: ‘Jointly Combining Implicit Constraints Improves Temporal Ordering’, in: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pp. 698–706, Association for Computational Linguistics, Stroudsburg, PA, USA, 2008.
- Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer T. Chayes, Levent Sagun, and Riccardo Zecchina: ‘Entropy-SGD: Biasing Gradient Descent Into Wide Valleys’, *CoRR* abs/1611.01838, 2016.
- Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa: ‘Natural Language Processing (Almost) from Scratch’, *J. Mach. Learn. Res.* 12: 2493–2537, November 2011.
- Savelie Cornegruta and Andreas Vlachos: ‘Timeline extraction using distant supervision and joint inference’, in: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pp. 1936–1942, 2016.

- Agata Cybulska and Piek Vossen: ‘Using a Sledgehammer to Crack a Nut? Lexical Diversity and Event Coreference Resolution’, in Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis (Eds.): *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC’14)*, European Language Resources Association (ELRA), Reykjavik, Iceland, may 2014.
- Donald Davidson: ‘The Individuation of Events’, in Nicholas Rescher (Ed.): *Essays in Honor of Carl G. Hempel*, pp. 216–34, Reidel, 1969.
- Thomas G. Dietterich: ‘Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms’, *Neural Computation* 10 (7): 1895–1923, October 1998.
- Quang Xuan Do, Wei Lu, and Dan Roth: ‘Joint Inference for Event Timeline Construction’, in: *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL ’12*, pp. 677–687, Association for Computational Linguistics, Stroudsburg, PA, USA, 2012.
- Timothy Dozat: ‘Incorporating Nesterov Momentum into Adam’, 2015.
- John Duchi, Elad Hazan, and Yoram Singer: ‘Adaptive Subgradient Methods for Online Learning and Stochastic Optimization’, *J. Mach. Learn. Res.* 12: 2121–2159, July 2011.
- Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio: ‘Why Does Unsupervised Pre-training Help Deep Learning?’, *Journal of Machine Learning Research* 11: 625–660, March 2010.
- Marieke van Erp, Piek Vossen, Rodrigo Agerri, Anne-Lyse Minard, Manuela Speranza, Ruben Urizar, Egoitz Laparra, Itziar Aldabe, and German Rigau: ‘Annotated Data, version 2’, *Technical report*, Amsterdam, Netherlands, 2015.
- Brian S. Everitt: *The analysis of contingency tables*, Halsted Press, New York, 1977.
- Julian J. Faraway: *Practical Regression and ANOVA using R*, University of Bath, 2002.
- Charles J. Fillmore: ‘Frame semantics and the nature of language’, in S. Harnad (Ed.): *Origins and evolution of language and speech*, pp. 155–202, Academy of Sciences, 1976.
- Charles J. Fillmore: *Frame semantics*, pp. 111–137, Hanshin Publishing Co., Seoul, South Korea, 1982.
- Owen Flanagan: *Consciousness Reconsidered*, MIT Press, 1992.

- Yarin Gal and Zoubin Ghahramani: ‘A Theoretically Grounded Application of Dropout in Recurrent Neural Networks’, in: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 1019–1027, 2016.
- Geoffrey Hinton: ‘Neural Networks for Machine Learning - Lecture 6a - Overview of mini-batch gradient descent’, 2012.
- Sepp Hochreiter and Jürgen Schmidhuber: ‘Flat Minima’, *Neural Computation* 9 (1): 1–42, January 1997b.
- Sepp Hochreiter and Jürgen Schmidhuber: ‘Long Short-Term Memory’, *Neural Computation* 9 (8): 1735–1780, November 1997a.
- Zhiheng Huang, Wei Xu, and Kai Yu: ‘Bidirectional LSTM-CRF Models for Sequence Tagging’, *CoRR* abs/1508.01991, 2015.
- Frank Hutter, Holger Hoos, and Kevin Leyton-Brown: ‘An Efficient Approach for Assessing Hyperparameter Importance’, in: *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32, ICML’14*, pp. I-754–I-762, JMLR.org, 2014.
- Alex Judea and Michael Strube: ‘Event Extraction as Frame-Semantic Parsing’, in: *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics (*SEM 2015)*, pp. 159–164, Association for Computational Linguistics, 2015.
- Daniel Jurafsky and James H. Martin: *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition, 2009.
- Nattiya Kanhabua and Kjetil Nørvåg: ‘Learning to Rank Search Results for Time-sensitive Queries’, in: *Proceedings of the 21st ACM International Conference on Information and Knowledge Management, CIKM ’12*, pp. 2463–2466, ACM, New York, NY, USA, 2012.
- Kenji Kawaguchi: ‘Deep Learning without Poor Local Minima’, in D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Eds.): *Advances in Neural Information Processing Systems 29*, pp. 586–594, Curran Associates, Inc., 2016.
- Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang: ‘On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima’, *arXiv preprint arXiv:1609.04836* 2016.
- Jaegwon Kim: *Supervenience and Mind - Events as Property Exemplifications*, Cambridge University Press, 1993.
- Diederik P. Kingma and Jimmy Ba: ‘Adam: A Method for Stochastic Optimization’, *CoRR* abs/1412.6980, 2014.

- Oleksandr Kolomiyets, Steven Bethard, and Marie-Francine Moens: ‘Extracting Narrative Timelines As Temporal Dependency Structures’, in: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL ’12, pp. 88–97, Association for Computational Linguistics, Stroudsburg, PA, USA, 2012.
- Alexandros Komninos and Suresh Manandhar: ‘Dependency Based Embeddings for Sentence Classification Tasks’, in: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1490–1500, Association for Computational Linguistics, San Diego, California, June 2016.
- Klaus Krippendorff: *Content Analysis: An Introduction to Its Methodology (second edition)*, Sage Publications, 2004.
- Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer: ‘Neural Architectures for Named Entity Recognition’, in: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 260–270, Association for Computational Linguistics, 2016.
- J. Richard Landis and Gary G. Koch: ‘The Measurement of Observer Agreement for Categorical Data’, *Biometrics* 33 (1): 159–174, 1977.
- Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel: ‘Backpropagation Applied to Handwritten Zip Code Recognition’, *Neural Computation* 1 (4): 541–551, December 1989.
- Yann Lecun: *Generalization and network design strategies*, Elsevier, 1989.
- Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller: ‘Efficient BackProp’, in: *Neural Networks: Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop*, pp. 9–50, Springer-Verlag, London, UK, UK, 1998.
- Heeyoung Lee, Marta Recasens, Angel Chang, Mihai Surdeanu, and Dan Jurafsky: ‘Joint Entity and Event Coreference Resolution Across Documents’, in: *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL ’12, pp. 489–500, Association for Computational Linguistics, Stroudsburg, PA, USA, 2012.
- Omer Levy and Yoav Goldberg: ‘Dependency-Based Word Embeddings’, in: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 2: Short Papers*, pp. 302–308, 2014.
- Qi Li, Heng Ji, and Liang Huang: ‘Joint Event Extraction via Structured Prediction with Global Features’, in: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 73–82, Association for Computational Linguistics, Sofia, Bulgaria, August 2013.

- Linguistic Data Consortium: *DEFT ERE Annotation Guidelines: Events*, v1.1 edition, 2013.
- Linguistic Data Consortium: *DEFT Rich ERE Annotation Guidelines: Events*, v2.5.1 edition, 2015.
- Liyuan Liu, Jingbo Shang, Frank F. Xu, Xiang Ren, Huan Gui, Jian Peng, and Jiawei Han: ‘Empower Sequence Labeling with Task-Aware Neural Language Model’, *CoRR* abs/1709.04109, 2017a.
- Shulin Liu, Yubo Chen, Shizhu He, Kang Liu, and Jun Zhao: ‘Leveraging FrameNet to Improve Automatic Event Detection’, in: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2134–2143, Association for Computational Linguistics, 2016.
- Shulin Liu, Yubo Chen, Kang Liu, and Jun Zhao: ‘Exploiting Argument Information to Improve Event Detection via Supervised Attention Mechanisms’, in: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1789–1798, Association for Computational Linguistics, 2017b.
- Xuezhe Ma and Eduard H. Hovy: ‘End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF’, *CoRR* abs/1603.01354, 2016.
- Inderjeet Mani, Marc Verhagen, Ben Wellner, Chong Min Lee, and James Pustejovsky: ‘Machine Learning of Temporal Relations’, in: *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, pp. 753–760, 2006.
- Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini: ‘Building a Large Annotated Corpus of English: The Penn Treebank’, *Comput. Linguist.* 19 (2): 313–330, June 1993.
- Frank J. Massey: ‘The Kolmogorov-Smirnov Test for Goodness of Fit’, *Journal of the American Statistical Association* 46 (253): 68–78, 1951.
- Gábor Melis, Chris Dyer, and Phil Blunsom: ‘On the State of the Art of Evaluation in Neural Language Models’, *CoRR* abs/1707.05589, 2017.
- Tomáš Mikolov: *Statistical language models based on neural networks*, Ph.D. thesis, Brno University of Technology, 2012.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean: ‘Efficient Estimation of Word Representations in Vector Space’, *CoRR* abs/1301.3781, 2013.
- Anne-Lyse Minard, Manuela Speranza, Eneko Agirre, Itziar Aldabe, Marieke van Erp, Bernardo Magnini, German Rigau, and Ruben Urizar: ‘SemEval-2015 Task 4: TimeLine: Cross-Document Event Ordering’, in: *Proceedings of the 9th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2015, Denver, Colorado, USA, June 4-5, 2015*, pp. 778–786, 2015.

- Anne-Lyse Minard, Manuela Speranza, Ruben Urizar, Begoña Altuna, Marieke van Erp, Anneleen Schoen, and Chantal van Son: ‘MEANTIME, the NewsReader Multilingual Event and Time Corpus’, in Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis (Eds.): *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, European Language Resources Association (ELRA), Paris, France, may 2016.
- Teruko Mitamura, Zhengzhong Liu, and Eduard Hovy: ‘Overview of TAC-KBP 2015 Event Nugget Track’, in: *Proceedings of the TAC-KBP 2015 Workshop*, Gaithersburg, Maryland, USA, 2015a.
- Teruko Mitamura, Yukari Yamakawa, Susan Holm, Zhiyi Song, Ann Bies, Seth Kulick, and Stephanie Strassel: ‘Event Nugget Annotation: Processes and Issues’, in: *Proceedings of the 3rd Workshop on EVENTS at the NAACL-HLT 2015*, pp. 66–76, Association for Computational Linguistics, June 2015b.
- Yurii Nesterov: ‘A method of solving a convex programming problem with convergence rate $O(1/\sqrt{k})$ ’, *Soviet Mathematics Doklady* 27: 372–376, 1983.
- Marta Guerrero Nieto, Roser Saurí, and Miguel Ángel Bernabé Poveda: ‘ModeS TimeBank: A Modern Spanish TimeBank Corpus’, *Procesamiento del Lenguaje Natural* 47: 259–267, 2011.
- Qiang Ning, Zhili Feng, and Dan Roth: ‘A Structured Learning Approach to Temporal Relation Extraction’, in: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 1038–1048, Association for Computational Linguistics, 2017.
- Tim O’Gorman, Kristin Wright-Bettner, and Martha Palmer: ‘Richer Event Description: Integrating event coreference with temporal, causal and bridging annotation’, in: *Proceedings of 2nd Workshop on Computing News Storylines*, pp. 47–56, Association for Computational Linguistics, November 2016.
- Martha Palmer, Will Styler, Kevin Crooks, and Tim O’Gorman: ‘Richer Event Description (RED) Annotation Guidelines (v.1.7)’, 2016, Online: <https://github.com/timjogorman/RicherEventDescription/blob/master/guidelines.md>.
- Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio: ‘On the Difficulty of Training Recurrent Neural Networks’, in: *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML’13, pp. III–1310–III–1318, JMLR.org, 2013.
- Jeffrey Pennington, Richard Socher, and Christopher D. Manning: ‘GloVe: Global Vectors for Word Representation’, in: *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.
- Sameer S. Pradhan, Lance A. Ramshaw, Ralph M. Weischedel, Jessica MacBride, and Linnea Micciulla: ‘Unrestricted Coreference: Identifying Entities and Events

- in OntoNotes.’, in: *Proceedings of the IEEE International Conference on Semantic Computing (ICSC)*, pp. 446–453, IEEE Computer Society, September 2007.
- James Pustejovsky: ‘The Syntax of Event Structure’, *Cognition* 41 (1991) pp. 47–81, 1991.
- James Pustejovsky, Patrick Hanks, Roser Sauri, Andrew See, Robert Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, and Marcia Lazo: ‘The TIMEBANK Corpus’, in: *Proceedings of Corpus Linguistics 2003*, pp. 647–656, Lancaster, UK, 2003.
- Nils Reimers, Philip Beyer, and Iryna Gurevych: ‘Task-Oriented Intrinsic Evaluation of Semantic Textual Similarity’, in: *Proceedings of the 26th International Conference on Computational Linguistics (COLING)*, pp. 87–96, December 2016a.
- Nils Reimers, Nazanin Dehghani, and Iryna Gurevych: ‘Temporal Anchoring of Events for the TimeBank Corpus’, in: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, Vol. 1: Long Papers, pp. 2195–2204, Association for Computational Linguistics, August 2016b.
- Nils Reimers, Nazanin Dehghani, and Iryna Gurevych: ‘Event Time Extraction with a Decision Tree of Neural Classifiers’, *Transactions of the Association for Computational Linguistics* 6: 77–89, 2018.
- Nils Reimers, Judith ECKLE-KOHLER, Carsten Schnober, Jungi Kim, and Iryna Gurevych: ‘GermEval-2014: Nested Named Entity Recognition with Neural Networks’, in Gertrud Faaß and Josef Ruppenhofer (Eds.): *Workshop Proceedings of the 12th Edition of the KONVENS Conference*, pp. 117–120, Universitätsverlag Hildesheim, October 2014.
- Nils Reimers and Iryna Gurevych: ‘Event Nugget Detection, Classification and Coreference Resolution using Deep Neural Networks and Gradient Boosted Decision Trees’, in: *Proceedings of the Eighth Text Analysis Conference (TAC 2015)*, National Institute of Standards and Technology (NIST), January 2015.
- Nils Reimers and Iryna Gurevych: ‘Optimal Hyperparameters for Deep LSTM-Networks for Sequence Labeling Tasks’, *arXiv preprint arXiv:1707.06799* 2017b.
- Nils Reimers and Iryna Gurevych: ‘Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging’, in: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 338–348, Copenhagen, Denmark, September 2017a.
- Nils Reimers and Iryna Gurevych: ‘Why Comparing Single Performance Scores Does Not Allow to Draw Conclusions About Machine Learning Approaches’, *CoRR* abs/1803.09578, 2018, Online: <http://arxiv.org/abs/1803.09578>.
- Stefan Riezler and John T. Maxwell: ‘On Some Pitfalls in Automatic Evaluation and Significance Testing for MT’, in: *Proceedings of the ACL Workshop on Intrinsic*

- and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pp. 57–64, Association for Computational Linguistics, Ann Arbor, Michigan, June 2005.
- David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams: ‘Learning representations by back-propagating errors’, *Nature* 323: 533–536, October 1986.
- Cícero Nogueira dos Santos, Bing Xiang, and Bowen Zhou: ‘Classifying Relations by Ranking with Convolutional Neural Networks’, in: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26–31, 2015, Beijing, China, Volume 1: Long Papers*, pp. 626–634, 2015.
- Roser Sauri and Toni Badia: ‘Spanish TimeBank 1.0’, LDC Catalog No.: LDC2012T12, 2012.
- Roser Saurí, Jessica Littman, Robert Gaizauskas, Andrea Setzer, and James Pustejovsky: ‘TimeML Annotation Guidelines, Version 1.2.1’, 2004, Online: http://timeml.org/site/publications/timeMLdocs/annguide_1.2.1.pdf.
- Andrea Setzer: *Temporal Information in Newswire Articles: An Annotation Scheme and Corpus Study*, Ph.D. thesis, University of Sheffield, Sheffield, UK, 2001.
- Jasper Snoek, Hugo Larochelle, and Ryan P Adams: ‘Practical Bayesian Optimization of Machine Learning Algorithms’, in F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Eds.): *Advances in Neural Information Processing Systems 25*, pp. 2951–2959, Curran Associates, Inc., 2012.
- Zhiyi Song, Ann Bies, Stephanie Strassel, Joe Ellis, Teruko Mitamura, Hoa Dang, Yukari Yamakawa, and Sue Holm: ‘Event Nugget and Event Coreference Annotation’, in: *Proceedings of the 4th Workshop on Events: Definition, Detection, Coreference, and Representation*, pp. 37–45, Association for Computational Linguistics, June 2016.
- Zhiyi Song, Ann Bies, Stephanie Strassel, Tom Riese, Justin Mott, Joe Ellis, Jonathan Wright, Seth Kulick, Neville Ryant, and Xiaoyi Ma: ‘From Light to Rich ERE: Annotation of Entities, Relations, and Events’, in: *Proceedings of the 3rd Workshop on EVENTS at the NAACL-HLT 2015*, pp. 89–98, Association for Computational Linguistics, June 2015.
- Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov: ‘Dropout: A Simple Way to Prevent Neural Networks from Overfitting’, *J. Mach. Learn. Res.* 15 (1): 1929–1958, January 2014a.
- Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov: ‘Dropout: A Simple Way to Prevent Neural Networks from Overfitting’, *J. Mach. Learn. Res.* 15 (1): 1929–1958, January 2014b.
- Angus Stevenson: *Oxford Dictionary of English* -, OUP Oxford, New York, London, 2010.

- Jannik Strötgen and Michael Gertz: ‘A Baseline Temporal Tagger for all Languages’, in: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 541–547, Association for Computational Linguistics, Lisbon, Portugal, September 2015.
- William F. Styler, Steven Bethard, Sean Finan, Martha Palmer, Sameer Pradhan, Piet C. de Groen, Bradley James Erickson, Timothy A. Miller, Chen Lin, Guer-gana K. Savova, and James Pustejovsky: ‘Temporal Annotation in the Clinical Domain’, *Transactions of the Association of Computational Linguistics* 2: 143–154, 2014.
- Mihai Surdeanu: ‘Overview of the TAC 2013 Knowledge Base Population Evaluation: English Slot Filling and Temporal Slot Filling’, in: *Proceedings of the TAC-KBP 2013 Workshop*, Gaithersburg, Maryland, USA, 2013.
- Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich: ‘Going Deeper with Convolutions’, in: *Computer Vision and Pattern Recognition (CVPR)*, 2015.
- Sara Tonelli, Rachele Sprugnoli, Manuela Speranza, and Anne-Lyse Minar: ‘NewsReader Guidelines for Annotation at Document Level’, *Technical Report NWR-2014-2-2*, Fondazione Bruno Kessler, 2014, Online: <http://www.newsreader-project.eu/files/2014/12/NWR-2014-2-2.pdf>.
- Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer: ‘Feature-rich Part-of-speech Tagging with a Cyclic Dependency Network’, in: *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL 2003, pp. 173–180, Association for Computational Linguistics, Stroudsburg, PA, USA, 2003.
- Naushad UzZaman and James F. Allen: ‘TRIPS and TRIOS System for TempEval-2: Extracting Temporal Information from Text’, in: *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval ’10, pp. 276–283, Association for Computational Linguistics, Stroudsburg, PA, USA, 2010.
- Naushad UzZaman, Hector Llorens, Leon Derczynski, Marc Verhagen, James F. Allen, and James Pustejovsky: ‘SemEval-2013 Task 1: TempEval-3: Evaluating Time Expressions, Events, and Temporal Relations’, in: *Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval 2013)*, pp. 1–9, Atlanta, Georgia, USA, 2013.
- Sudhir Varma and Richard Simon: ‘Bias in error estimation when using cross-validation for model selection.’, *BMC bioinformatics* 7 (1): 91–99, February 2006.
- Marc Verhagen, Robert Gaizauskas, Frank Schilder, Mark Hepple, Graham Katz, and James Pustejovsky: ‘SemEval-2007 Task 15: TempEval Temporal Relation Identification’, in: *Proceedings of the 4th International Workshop on Semantic Evaluations*, SemEval ’07, pp. 75–80, Association for Computational Linguistics, Stroudsburg, PA, USA, 2007.

- Marc Verhagen, Roser Saurí, Tommaso Caselli, and James Pustejovsky: ‘SemEval-2010 Task 13: TempEval-2’, in: *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval ’10, pp. 57–62, Association for Computational Linguistics, Stroudsburg, PA, USA, 2010.
- Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda: ‘ACE 2005 Multilingual Training Corpus’, LDC Catalog No.: LDC2006T06, 2005.
- Bishan Yang and Tom Mitchell: ‘Joint Extraction of Events and Entities within a Document Context’, in: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 289–299, 2016.
- Seid Muhie Yimam, Iryna Gurevych, Richard Eckart de Castilho, and Chris Biemann: ‘WebAnno: A Flexible, Web-based and Visually Supported System for Distributed Annotations’, in: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 1–6, Association for Computational Linguistics, Sofia, Bulgaria, August 2013.
- Katsumasa Yoshikawa, Sebastian Riedel, Masayuki Asahara, and Yuji Matsumoto: ‘Jointly Identifying Temporal Relations with Markov Logic’, in: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1*, ACL ’09, pp. 405–413, Association for Computational Linguistics, Stroudsburg, PA, USA, 2009.
- Matthew D. Zeiler: ‘ADADELTA: An Adaptive Learning Rate Method’, *CoRR* abs/1212.5701, 2012.
- Daniel Zeman, Martin Popel, Milan Straka, Jan Hajic, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, Francis Tyers, Elena Badmaeva, Memduh Gokirmak, Anna Nedoluzhko, Silvie Cinkova, Jan Hajic jr., Jaroslava Hlavacova, Václava Kettnerová, Zdenka Uresova, Jenna Kanerva, Stina Ojala, Anna Missilä, Christopher D. Manning, Sebastian Schuster, Siva Reddy, Dima Taji, Nizar Habash, Herman Leung, Marie-Catherine de Marnette, Manuela Sanguinetti, Maria Simi, Hiroshi Kanayama, Valeria dePaiva, Kira Drohanova, Héctor Martínez Alonso, Çağrı Çöltekin, Umut Sulubacak, Hans Uszkoreit, Vivien Macketanz, Aljoscha Burchardt, Kim Harris, Katrin Marheinecke, Georg Rehm, Tolga Kayadelen, Mohammed Attia, Ali Elkahky, Zhuoran Yu, Emily Pitler, Saran Lertpradit, Michael Mandl, Jesse Kirchner, Hector Fernandez Alcalde, Jana Strnadová, Esha Banerjee, Ruli Manurung, Antonio Stella, Atsuko Shimada, Sookyoung Kwak, Gustavo Mendonca, Tatiana Lando, Rattima Nitisaroj, and Josie Li: ‘CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies’, in: *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pp. 1–19, Association for Computational Linguistics, Vancouver, Canada, August 2017.
- Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao: ‘Relation Classification via Convolutional Deep Neural Network’, in: *COLING 2014, 25th*

BIBLIOGRAPHY

International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, August 23-29, 2014, pp. 2335–2344, Dublin, Ireland, 2014.